Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Mestrado em Engenharia Informática

PIPE: Uma infra-estrutura genérica para ambientes de computação ubíqua

Bruno Oliveira Félix(26265)

Lisboa
(2009)

Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

PIPE: Uma infra-estrutura genérica para ambientes de computação ubíqua

Bruno Oliveira Félix (26265)

Orientador: Prof. Doutor Nuno Manuel Ribeiro Preguiça

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para a obtenção do Grau de Mestre em Engenharia Informática.

Lisboa
(2009)

Aos meus pais

Agradecimentos

Em primeiro lugar gostaria de agradecer ao Centro de Informática e Tecnologias da Informação da FCT-UNL, pela bolsa de investigação concedida.

Em particular, gostava de agradecer ao meu orientador, Nuno Manuel Preguiça, não só pela orientação e conselhos imprescindíveis, mas também pelo seu entusiasmo e dedicação ao longo da realização deste trabalho.

Queria também agradecer aos meus amigos e companheiros de jornada, Ana Lameira, David Navalho, Hélio Dolores, Nuno Luís, e Simão Mata, pelo seu apoio, ideias, comentários, boa disposição e noitadas patrocinadas pela Maria Café. Foi um privilégio contar com a sua companhia.

Ao professor João Lourenço, pelas conversas interessantes e sugestões de leitura.

Aos meus pais e aos meus irmãos Nuno e Sofia, pela paciência, compreensão e incentivo, mostrados ao longo de todo o meu percurso académico.

Por fim, não posso deixar de agradecer aos demais amigos, e em especial à Inês pelas noites de cultura, caminhadas revigorantes e boa companhia.

A todos, muito obrigado.

Summary

In the last decades we have witnessed a significant growth in the number and power of computing devices. These can be divided in two broad categories. First, the devices that pervade most of the spaces we use, and second the mobile equipments carried by the users in their day-to-day life.

However, up until recently both these classes of devices coexisted without much interaction. It is therefore interesting to explore architectures that integrate these different devices, in order to leverage their strong points, and mask some of their inherent shortcomings.

In this dissertation, we present an infrastructure aimed at public spaces that allows users equipped with mobile devices, to take advantage of the provided resources (e.g. shared displays, computing power). This system, designated PIPE (Pervasive Infrastructure for Public Environments), provides different pre-defined applications to the user. Additionally, it also allows the user to execute his own applications in the infrastructure, extending the capabilities of his mobile device.

PIPE presents a service oriented architecture that allows the usage of several applications, whose execution might be distributed between the infrastructure and the available mobile devices. These applications are integrated in a pervasive computing environment, that grants them access to the provided resources, encouraging their rational usage.

The results obtained during the evaluation of this system, show that the distributed execution of applications, in which the most resource-intensive parts are executed in the infrastructure, allows for significative performance improvements.

Keywords: pervasive computing, mobile devices, shared displays, cyber foraging.

Sumário

Nas últimas décadas tem-se assistido a um crescimento significativo do número e poder dos diferentes dispositivos computacionais. Este crescimento traduz-se na coexistência de duas grandes classes de equipamentos. Por um lado, aqueles que permeiam de forma quase ubíqua os espaços que frequentamos e, por outro lado, os equipamentos móveis que acompanham os utilizadores no seu dia-a-dia.

No entanto, estas duas classes de dispositivos têm existido em mundos largamente separados. Este facto torna interessante a exploração de modelos e arquitecturas que permitam tirar partido das melhores características de cada uma delas, mascarando as suas limitações.

Nesta dissertação apresenta-se um sistema ubíquo vocacionado para espaços públicos, que permite a utilizadores equipados com dispositivos móveis usufruir dos recursos disponibilizados (e.g., ecrãs partilhados, capacidade computacional). Este sistema, designado de PIPE (Pervasive Infrastructure for Public Environments), oferece um conjunto aplicações pré-definidas a que o utilizador pode aceder. Adicionalmente, permite que o utilizador execute as suas próprias aplicações na infra-estrutura, estendendo a capacidade dos dispositivos portáteis.

O PIPE apresenta uma arquitectura orientada aos serviços, que permite acomodar um conjunto vasto de aplicações, podendo a sua execução ser distribuída entre a infra-estrutura e os dispositivos móveis presentes. Estas aplicações ficam integradas num ambiente de computação ubíqua, que lhes faculta o acesso aos diferentes recursos disponibilizados, promovendo uma utilização racional dos mesmos.

Os resultados obtidos durante a avaliação do PIPE, mostram que a execução distribuída de aplicações, com as componentes mais exigentes das mesmas a executar na infra-estrutura, permite melhorias de desempenho significativas.

Palavras-chave: computação ubíqua, dispositivos móveis, ecrãs partilhados, execução remota.

Conteúdo

Lista de Figuras

Lista de Tabelas

1

Introdução

1.1 Contexto

Nos últimos quarenta anos têm-se assistido a avanços muito significativos no campo da electrónica, permitindo a criação de dispositivos computacionais, cada vez mais pequenos, mais potentes e mais baratos.

Isto permite que muitos espaços públicos, como centros comerciais, estações de caminhos de ferro ou aeroportos sejam cada vez mais permeados por um conjunto muito alargado de equipamentos (computadores, ecrãs, projectores, etc).

Também no campo das telecomunicações tem havido uma evolução constante, permitindo que diferentes tipos de dispositivos sejam equipados com um crescente número de tecnologias de comunicação sem fios (GSM [gsm09], Wi-Fi [wif09], Bluetooth [blu09], IRDA [ird09]). Estas tecnologias estão também presentes em muitos dos espaços que frequentamos, tendo havido uma grande proliferação de redes GSM/3G ou Wi-Fi, aumentando a conectividade à rede em qualquer lugar.

Uma área onde os avanços nestes dois campos têm sido particularmente notórios é na área dos dispositivos móveis, sendo a sua generalização a grande parte da população uma realidade [mob09]. O crescente número de tecnologias de comunicação integrado nestes dispositivos permite que em muitas situações estes sejam utilizados como *proxies* para o utilizador [PKH+06] e como meios para interagir com outros sistemas.

1.2 Motivação

Neste contexto, torna-se por isso relativamente comum encontrar num espaço vários ecrãs (e outro hardware), muitas vezes sub-aproveitados [HKB08], ao mesmo tempo que há uma grande quantidade de utilizadores munidos de dispositivos móveis (telemóveis, PDAs). Surge por isso a oportunidade de tirar partido das melhores características destas duas classes de dispositivos, em particular a conectividade, resolução e poder computacional dos equipamentos de infra-estrutura e a crescente capacidade de armazenamento e personalização associada aos dispositivos pessoais dos utilizadores.

A integração dos dispositivos pessoais dos utilizadores de um espaço, em ambientes cada vez mais permeados por equipamentos com capacidade computacional, aproxima-nos da visão de computação ubíqua de Weiser [Wei99]. Esta integração permite criar novas oportunidades de interacção quer entre dispositivos, quer entre utilizadores do mesmo espaço, possibilitando a criação de novas aplicações que possam usufruir dos recursos e serviços oferecidos.

Neste contexto, existem já algumas aplicações que mostram o potencial desta abordagem. No âmbito da publicidade direccionada por exemplo, a utilização combinada de recursos presentes na infra-estrutura, com os dispositivos pessoais dos utilizadores apresenta bastante interesse [AGKO04, KPD07]. Isto permite a criação de aplicações que mostram publicidade de acordo com as preferências dos utilizadores, e com base nos padrões de actividade detectados. Os anúncios mostrados podem ser mais que conteúdos estáticos, dando ao utilizador a possibilidade de efectuar imediatamente a compra, ou saber mais informação sobre o produto ou serviço anunciado.

Um espaço desta natureza, também poderia possibilitar a criação de outras aplicações. Por exemplo, num centro comercial, seria interessante ter um serviço de informação sobre as lojas presentes, facultando ao utilizador informação relativa ao seu horário de funcionamento, localização e produtos disponíveis (e.g. [DSP08]). Outros serviços interessantes poderiam incluir o suporte para armazenamento e replicação de dados, nos moldes apresentados por Shapiro et. al em [BFS07].

Levando esta visão um passo adiante, um espaço desta natureza deverá possibilitar interacções fortuitas, mostrando ao utilizador novas informações e serviços de forma não intrusiva, ao mesmo tempo que permite ao utilizador levar um estilo de vida móvel, tirando partido da infra-estrutura existente para correr as aplicações e serviços pretendidos.

1.3 Objectivos

Nesta dissertação apresenta-se um sistema que permite oferecer um espaço inteligente, que permite aos utilizadores integrar os seus dispositivos pessoais num ambiente de computação, de forma a tirar partido dos recursos e funcionalidades oferecidos.

Seguidamente apresentam-se brevemente os principais objectivos que guiaram a criação da solução apresentada neste documento.

Custo: A implementação de uma arquitectura ubíqua para o fornecimento de serviços não deve comportar um custo muito elevado. De preferência deve poder tirar partido de muito do hardware que já existe espalhado pelos diferentes locais, e também não deve exigir equipamento específico da parte dos clientes. Mesmo nestas condições, uma infra-estrutura deste tipo terá alguns custos. O estudo dos modelos de negócio associados à disponibilização deste tipo de infra-estrutura está fora do âmbito deste trabalho, mas poderia passar pela utilização de receitas de publicidade apresentada na infra-estrutura ou através de um modelo de partilha de custos com os utilizadores.

Flexibilidade: Um sistema desta natureza deve contemplar os mecanismos para fazer uma gestão fácil dos serviços que a infra-estrutura oferece, tornando o processo de adição de um novo serviço relativamente simples. A plataforma deve permitir que estes sejam obtidos a partir de diferentes meios e localizações, podendo ficar associados a localizações específicas.

Adicionalmente, o sistema deve também permitir ao utilizador executar as suas próprias aplicações ou serviços, tirando partido serviços e recursos disponibilizados pela infra-estrutura. O objectivo último é promover uma utilização mais racional dos dos mesmos, facilitando a criação de novas aplicações.

Tempo de resposta: Numa plataforma vocacionada para espaços públicos, como por exemplo centros comerciais, a questão do tempo de resposta no acesso aos serviços disponibilizados é bastante importante. Neste tipo de ambiente, não é expectável que o utilizador tenha disposição para esperar bastante tempo para interagir com a infra-estrutura, pelo que esse facto deverá ser considerado.

Segurança e auditabilidade: Em qualquer sistema a questão da segurança é importante. Para um sistema que permite a colaboração de equipamentos da infra-estrutura com dispositivos móveis dos utilizadores, e onde é dada a possibilidade destes últimos executarem na infra-estrutura as suas próprias aplicações, esta questão é ainda mais premente. Neste contexto existem diversos problemas distintos, dos quais se destacam a protecção da infra-estrutura face à execução de software arbitrário e o controlo de acessos de uma aplicação relativamente aos recursos que pode utilizar.

1.4 Âmbito do trabalho

Esta dissertação apresenta uma plataforma que permite conjugar os recursos fornecidos pela infra-estrutura com os equipamentos pessoais dos utilizadores.

Este sistema designado de PIPE (Pervasive Infrastructure for Public Environments), oferece um ambiente de computação ubíqua no qual as diferentes aplicações que nele executam podem ser integradas, tirando partido dos recursos e serviços disponibilizados.

O PIPE apresenta uma arquitectura orientada aos serviços, que permite acomodar diferentes tipologias de aplicação. Estas apresentam uma grande variabilidade, desde aplicações que correm exclusivamente na infra-estrutura, até aplicações que executam totalmente nos dispositivos pessoais dos utilizadores, passando pelas aplicações cuja execução pode ser distribuída entre estas duas entidades.

A nossa visão é que este sistema possa ser implementado num local público (e.g. centro comercial), onde existe à partida uma grande quantidade de dispositivos computacionais espalhados pelo espaço físico e um número significativo de utilizadores.

O PIPE permite oferecer um conjunto pré-definido de aplicações e serviços, que são disponibilizados pelo próprio sistema. Os utilizadores podem aceder a essas aplicações efectuando o seu carregamento dinâmico, ou através de uma interface web. Um exemplo de uma aplicação de infra-estrutura típica, pode ser uma aplicação que corra num espaço comercial e que disponibilize blogs associados às diferentes lojas. Estes podem ser consultados e actualizados pelos utilizadores, contribuindo para a criação de um espaço mais interactivo.

Adicionalmente o PIPE permite carregar dinamicamente aplicações do utilizador na infra-estrutura. Esta funcionalidade possibilita uma maior flexibilidade, não ficando o utilizador limitado apenas às aplicações disponibilizadas nativamente pela plataforma. Considere-se por exemplo, uma situação em que um utilizador aguarda pelo seu voo num terminal de aeroporto. Para passar o tempo este decide jogar um jogo. Havendo a infra-estrutura presente, este poderia ser carregado directamente a partir do dispositivo móvel do utilizador, ou a partir de um servidor remoto. Esta aplicação podia fazer uso de um ecrã público, sendo o seu controlo efectuado a partir do dispositivo pessoal do utilizador.

1.4.1 Principais contribuições

Esta dissertação apresenta o desenho do PIPE, um sistema que apresenta uma arquitectura orientada aos serviços, que permite a colaboração entre dispositivos móveis e equipamentos que integram a infra-estrutura. Este fornece um ambiente de computação

onde as diferentes aplicações podem ser integradas, e que permite que estas façam uso dos serviços que a plataforma oferece.

Quando comparado com trabalhos relacionados (e.g. [KBM+00, LJP+06, RRA08]), o PIPE apresenta algumas diferenças importantes, em particular devido ao facto deste suportar não só um conjunto pré-definido de aplicações e serviços nativos ao próprio sistema, mas também permitir ao utilizador executar, parcial ou totalmente as suas aplicações na infra-estrutura.

Os resultados obtidos mostram a viabilidade desta aproximação, exibindo valores interessantes em termos do tempo de resposta ao utilizador, bem como reduções significativas na quantidade de informação que é transmitida para o dispositivo móvel.

1.5 Estrutura do documento

Nesta secção dá-se uma visão geral da organização deste documento. Após este capítulo introdutório, no capítulo 2, analisa-se algum do trabalho relacionado efectuado no âmbito onde o sistema PIPE se insere. O capítulo 3 é apresenta arquitectura do sistema PIPE, e no capítulo 4 são explicitados os aspectos relativos à sua implementação. Nos capítulos 5 e 6, são apresentadas respectivamente as aplicações de demonstração criadas e os resultados obtidos no âmbito da avaliação do sistema. Finalmente, o capítulo 7, conclui esta dissertação com algumas considerações finais e propostas de trabalho futuro.

2

Trabalho relacionado:

Neste capítulo será abordado algum do trabalho já efectuado no âmbito onde o nosso sistema se insere. Inicialmente serão identificados os principais problemas que advêm de um cenário onde se pretende combinar dispositivos móveis com equipamentos presentes num determinado espaço. Fazendo seguidamente um levantamento de diversos sistemas e aplicações que são representativos do esforço já desenvolvido nesta área e que ilustram algumas respostas, parciais ou totais, aos problemas levantados.

Em traços gerais, os sistemas estudados podem-se dividir em duas grandes categorias, sistemas que permitem a interacção entre dispositivos móveis e ecrãs públicos e plataformas de sistemas ubíquos para fornecimento de serviços.

Na primeira categoria iremos agrupar o conjunto de aplicações, ou plataformas para a criação de aplicações (e.g. [MC07, PAB+04]), que permitem a comunicação e interacção entre os dispositivos pessoais dos utilizadores e ecrãs espalhados por um determinado espaço.

A segunda categoria irá englobar as arquitecturas genéricas para o fornecimento de serviços que foram estudadas. Estas caracterizam-se por permitir a interacção entre o utilizador e respectivo dispositivo móvel com vários meios presentes na infra-estrutura, não se limitando a ecrãs, fornecendo um conjunto de serviços que pode ser facilmente aumentado.

2.1 Desafios

Os sistemas de computação ubíqua exibem um conjunto muito variado de desafios [Sat01]. Em particular no cenário para o qual o PIPE está vocacionado, existem problemas [NCL+08, RC02] de várias ordens. Alguns destes problemas são bem conhecidos dos sistemas distribuídos como a segurança, a heterogeneidade e os efeitos negativos da latência [DKC05], outros de cariz mais específico como a captura e agregação de informação dependente de contexto, a combinação de diferentes meios para disponibilizar conteúdos ao utilizador e a própria medição da eficácia dos sistemas e da satisfação dos utilizadores.

Seguidamente serão detalhados estes problemas no âmbito da utilização de dispositivos móveis em conjunto com recursos fornecidos pela infra-estrutura.

A **privacidade** é uma preocupação crucial neste contexto. Em particular, devido ao facto de muitas das aplicações que se podem imaginar (e.g. publicidade), poderem tirar partido de informação privada do utilizador, as suas preferências, ou a sua localização. Como tal importa por um lado garantir que a informação recolhida é mantida de forma segura, sendo destruída a seu tempo, e que, se esta for recolhida por terceiros é o mais anónima possível. Por outro lado, importa dar ao utilizador a hipótese deste não ser detectado ou importunado pelas aplicações e dispositivos que existam num espaço.

Um aspecto importante neste contexto prende-se com a privacidade da informação mostrada em ecrãs públicos. Uma abordagem simples poderia passar por apresentar a informação sensível exclusivamente no dispositivo pessoal do utilizador. No entanto isto pode ser um pouco restritivo demais, existindo outras alternativas. Por exemplo, é possível tornar ilegíveis certas partes do conteúdo mostrado num ecrã público [BKN+05]. Quando o utilizador pretende ver a informação que está "obscurecida", escolhe essa região e o conteúdo é transferido para o visor do telemóvel. O grande desafio neste ponto, consiste em escolher e definir automaticamente o que é que constitui informação sensível. Essa definição pode ser feita usando input por parte do utilizador, ou então usando meios automáticos, filtrando por exemplo certas palavras ou distorcendo certas regiões de uma imagem (por exemplo caras numa foto) [BKN+05].

Com base neste tipo de técnicas, as diferentes zonas de um texto ou de uma imagem podem ser mostradas, ou não, consoante o espaço em que o utilizador se encontra.

A questão da **segurança**, tal como acontece noutras áreas é uma questão muito relevante, que tem um impacto decisivo na confiança que o utilizador final tem em qualquer sistema. É por isso importante abordar brevemente algumas soluções propostas em trabalho anterior.

Um aspecto bastante importante prende-se com a escolha do dispositivo, por exemplo o ecrã, com o qual o cliente interage. Tipicamente esta escolha é feita seleccionando o nome do dispositivo a partir de uma lista [DSP08]. Na prática esta abordagem não é muito intuitiva nem muito segura já que se podem mandar conteúdos sensíveis para outro lugar diferente do pretendido. Uma alternativa que tem sido explorada, tem sido a utilização de tecnologias de comunicação sem fios de curto alcance (e.g. IRDA ou NFC [nfc09]) para escolher qual o dispositivo a interagir [BSSW02, KBM+00].

Nesta abordagem, a conexão é efectuada por proximidade, sendo bastante intuitiva, pois o utilizador estabelece contacto com um dispositivo na infra-estrutura simplesmente deslocando-se até ele. Esta confere ainda algumas vantagens em termos de segurança, já que é difícil a um atacante interferir em canais de comunicação de curto alcance, sem que o utilizador legítimo se aperceba. Adicionalmente, durante o contacto inicial, podem ser trocadas chaves públicas ou outros segredos, que são a base para um contacto seguro efectuado através do canal principal de comunicação (e.g. Wi-Fi) com a entidade pretendida.

Outro aspecto importante, tem a ver com a verificação da integridade e identidade do software presente num dado dispositivo. O objectivo é informar o cliente móvel caso haja alguma irregularidade no software que existe, dando garantias ao utilizador de estar a interagir com uma plataforma de confiança. Por exemplo Garriss et. al. [GCB+08] apresentam uma solução que assenta na utilização um Trusted Platform Module (TPM)[1].

Após considerar os aspectos de segurança e privacidade, abordam-se de seguida aspectos funcionais e de arquitectura dos sistemas.

A questão da **latência** na apresentação da informação tem um papel importante neste contexto, já que importa que esta seja apresentada ao utilizador atempadamente e enquanto este se encontra em determinado local. Em alguns sistemas (e.g. [AGKO04]), verifica-se que em certas circunstâncias, os anúncios enviados para os dispositivos pessoais chegaram tarde demais, muitas vezes quando os utilizadores já se encontravam a uma distância considerável da loja referenciada.

A solução para este problema passa pela utilização de meios de comunicação mais imediatos. Por exemplo, em [AGKO04], os dispositivos dos utilizadores eram detectados através de bluetooth, sendo os anúncios enviados através de SMS. Esta solução apresenta uma latência inaceitável, eventualmente uma alternativa mais viável para este cenário poderia consistir no envio dos anúncios através de OBEX sobre bluetooth.

A **captura e agregação da informação** é uma dificuldade típica destes sistemas,

[1] O TPM é uma componente de hardware que fornece um conjunto de primitivas criptográficas, bem como alguma memória onde é possível guardar alguns dados sensíveis como por exemplo chaves.

havendo a necessidade de encontrar métricas adequadas para que a informação recolhida seja significativa. Ao mesmo tempo, do ponto de vista de um sistema que resida neste tipo de ambientes, não é possível contar exclusivamente com contribuições explícitas dos utilizadores pelo que esta recolha deve ser feita de forma autónoma, levantando eventualmente problemas de privacidade no processo [RC02].

Num meio rico em dispositivos, as possibilidades de interacção são inúmeras, e é possível **combinar um conjunto bastante alargado de meios** para veicular informação ao utilizador. Existem diversas questões que estão subjacentes a esta problemática. Em primeiro lugar, a escolha e descoberta dos meios a utilizar convém ser feita de forma rápida e intuitiva. Em segundo lugar há que resolver a questão da partilha de meios (e.g. ecrãs públicos), por vários utilizadores. Por um lado, estes meios podem constituir pontos de contenção e conflito, mas podem também abrir oportunidades interessantes para socialização [BIF+04].

Na questão da partilha de ecrãs públicos por múltiplas aplicações e utilizadores, existe já bastante trabalho efectuado. Por exemplo em [JP06], apresenta-se uma solução genérica para problemática do acesso concorrente a um ecrã por parte de múltiplas aplicações. O objectivo é abstrair as aplicações de aspectos relacionados com a gestão dos ecrãs, ficando toda essa questão sob o controlo do sistema.

Finalmente há que decidir que conteúdos mostrar em cada dispositivo, podendo os conteúdos "sensíveis"ser mostrados no dispositivo móvel e outros conteúdos mostrados num ecrã público, tal como a solução apresentada em [BKN+05].

O conjunto de meios que pode ser combinado não se limita a meios de input e output, podendo também consistir noutros recursos como capacidade de processamento ou armazenamento. Como tal, coloca-se também a questão da forma como as aplicações podem ser decompostas de maneira a que algumas das suas componentes possam correr em computadores que estejam disponíveis. Este tipo de divisão, possibilita que os dispositivos móveis executem aplicações que façam um uso intensivo de diversos recursos, trazendo as componentes mais exigentes dessas aplicações (em termos computacionais ou de comunicação) para a infra-estrutura.

Exitem várias soluções para este problema. Algumas, como aquela que é utilizada em [RRA08], fazem uma divisão estática das componentes que podem executar nos diferentes dispositivos, definindo à partida exactamente quais são aquelas passíveis de ser executadas nos dispositivos pessoais dos utilizadores.

Outras soluções como [BGSH07], vêm demonstrar a viabilidade de se fazer uma adaptação rápida de aplicações para este tipo de cenário. Esta baseia-se na noção de que para uma dada aplicação há apenas um número limitado de formas de particionar a sua execução entre o dispositivo móvel, e servidores presentes na infra-estrutura.

Neste trabalho, as diferentes estratégias de particionamento são complementadas por um sistema de execução que permite migrar as várias componentes da aplicação face a flutuações nas condições de operação da mesma.

Num cenário em que os equipamentos exibem uma grande heterogeneidade, a **adaptação** dos conteúdos e aplicações às características do ambiente e meios envolvidos é fundamental. Idealmente esta adaptação deve levar em linha de conta os diferentes recursos existentes (e.g. bateria, capacidade de processamento disponível, resolução, etc) de forma a poder efectuar uma utilização mais racional dos mesmos, mantendo ainda assim uma qualidade que seja aceitável para o utilizador final.

Existem diversas abordagens a esta problemática. Soluções como o MOWSER [BJA98], fazem uso de *proxies* que efectuam a adaptação dos conteúdos solicitados pelas aplicações a que estão associados. Esta adaptação é efectuada consoante os recursos disponiveis e parâmetros de qualidade de serviço especificados.

Outros sistemas, como o Odyssey [NSN+97], são vocacionados para a adaptação dinâmica de conteúdos. Este consiste numa versão modificada do sistema de operação NetBSD, que monitoriza os diferentes recursos e notifica as aplicações das alterações detectadas. Cada aplicação decide individualmente como reagir às variações nos recursos disponíveis, podendo pedir ao sistema para adaptar os conteúdos utilizados.

Esta abordagem permite acomodar uma grande diversidade de políticas de adaptação, consoante as preferências de cada aplicação, ficando o sistema apenas a cargo da monitorização dos recursos existentes e da adaptação dos conteúdos fornecidos.

Por último, as **medições da performance** do sistema e da satisfação dos utilizadores são aspectos fundamentais, em particular em certos domínios, como por exemplo no caso da publicidade [NCL+08]. No entanto a obtenção desta informação poderá não ser trivial. Consideremos por exemplo o caso em que um utilizador vê num ecrã um anúncio de um produto que lhe interessa, mas só efectua a compra no dia seguinte. O anúncio foi eficaz, mas é difícil fazer a associação com a compra.

2.2 Sistemas de interacção entre dispositivos móveis e ecrãs partilhados

Nesta secção será apresentado em detalhe um conjunto representativo dos sistemas de interacção entre dispositivos e ecrãs partilhados estudados. O estudo destes sistemas permitiu ter uma noção mais clara das diferentes tipologias de aplicação, que podem fazer uso de um dispositivo móvel e dos recursos presentes na infra-estrutura.

Fundamentalmente, identificam-se três grandes tipologias de aplicação. Em primeiro lugar, aplicações que correm exclusivamente na infra-estrutura, podendo inclusiva-

mente não requerer uma interacção explícita com os utilizadores ou os seus dispositivos (e.g. [KPD07]).

Em segundo lugar, aplicações cuja execução pode ser distribuídas entre o cliente móvel e a infra-estrutura. Esta categoria é aquela que apresenta uma maior complexidade, tendo sido por isso estudado um conjunto mais vasto de sistemas [AGKO04, PAB+04, MC07, TSN07].

E por fim, aplicações que executam exclusivamente no dispositivo pessoal do utilizador e que podem eventualmente fazer uso de alguns recursos que a infra-estrutura forneça, como é o caso do [CKM+08].

2.2.1 Bluscreen

O Bluscreen [PDJS06, SPD06] é uma plataforma vocacionada para a apresentação de publicidade ou conteúdos informativos, em ecrãs públicos. Os anúncios são escolhidos de forma "inteligente", tendo em conta a audiência, de forma a maximizar a exposição de novos anúncios.

Arquitectura e funcionamento

O sistema Bluscreen visa tirar partido da massificação de dispositivos móveis com conectividade bluetooth que tem ocorrido ao longo dos últimos anos. Como tal, detecta os dispositivos equipados com esta tecnologia de comunicação que estejam nas imediações do ecrã, sendo cada dispositivo associado a um utilizador, ou seja, utiliza-se o dispositivo bluetooth como um proxy para um humano.

Este sistema é composto por três tipos de entidades distintas, o *device detection agent*, o *advertising agent* e o *marketplace agent* (fig. 2.1).

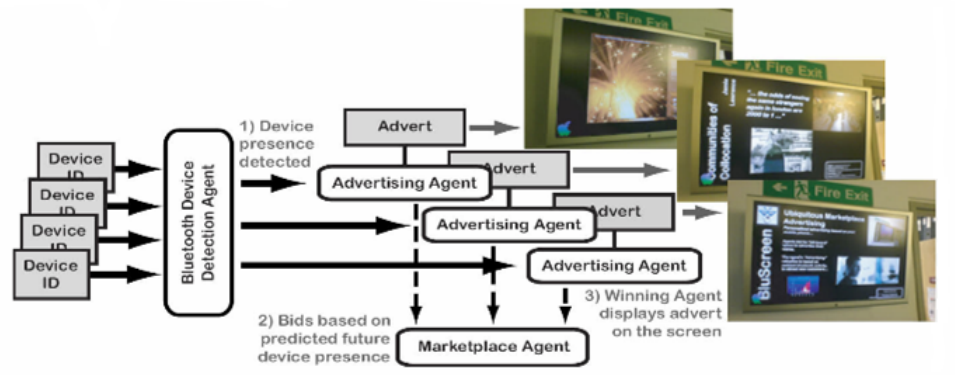

Figura 2.1: Arquitectura do sistema Bluscreen (de [KPD07])

Cada ecrã está associado a um *device detection agent* que no fundo é um sensor

bluetooth, que é responsável por detectar os dispositivos presentes numa vizinhança próxima.

Neste sistema cada anúncio é representado por um *advertising agent*, e tem um determinado "orçamento", que é gasto à medida que é escolhido para um dado ecrã. Quando este orçamento é esgotado, o anúncio é descartado.

A escolha dos anúncios é feita por meio de um sistema de leilões. A cada iteração deste processo, a componente de detecção informa os *advertising agents* dos utilizadores presentes, e estes propõem um valor para cada ecrã. Uma política razoável nesta situação consiste em valorizar mais os utilizadores que nunca viram o anúncio, de forma a garantir uma maior exposição de anúncios novos. O *marketplace agent* é responsável por agregar todas as propostas, e escolher para cada ecrã qual o anúncio que propõe o valor mais elevado.

Aspectos interessantes

Este trabalho apresenta alguns aspectos interessantes. Por um lado é um bom exemplo de uma aplicação que não requerendo software específico por parte dos utilizadores, consegue adaptar os conteúdos mostrados em função das características do público-alvo, funcionado em situações em que à partida não se conhece os interesses e tipologia dos utilizadores.

Tem ainda algum potencial para ser usado para inferir mais informação sobre o perfil dos utilizadores, tendo em conta os padrões de actividade detectados. Isto podia ser feito tendo em conta a localização do ecrã e de pontos de interesse próximos, por exemplo lojas, de forma a ter uma noção do percurso que um utilizador efectuou e dessa forma adequar melhor os conteúdos mostrados.

No sistema desenvolvido, permite-se a implementação de um sistema deste tipo.

2.2.2 Manhattan Story Mashups

O Manhattan Story Mashups (MSM) [TSN07], é um jogo que combina uma componente baseada na web, utilizadores em Manhattan dotados de telefones móveis equipados com câmaras e um dos ecrãs presentes em Times Square. O objectivo é que os jogadores criem e ilustrem histórias que serão mostradas no ecrã público. Este jogo introduz uma forma colaborativa de narrar uma história possibilitando a cooperação entre os indivíduos "no terreno"e os jogadores ligados à rede.

Arquitectura e funcionamento

Este sistema é composto por três componentes, uma *storytelling tool*, um cliente móvel, e um ecrã de grandes dimensões num local de grande visibilidade.

A *storytelling tool* é uma aplicação web, responsável por oferecer um *front-end* para os jogadores submeterem uma história. Adicionalmente, coordena as comunicações com os jogadores no terreno, enviando-lhes quando necessário novas palavras.

O cliente móvel, consiste numa pequena aplicação escrita em Python para S60, que corre no dispositivo móvel, e é essencialmente responsável por receber novas palavras para o utilizador "anotar", e enviar a respectiva foto de volta ao servidor. Este tem ainda um terceiro modo de actuação, no qual o utilizador, dada uma imagem escolhe de uma lista de palavras aquela que mais se adequa, o chamado *guessing mode*.

O jogo em si é composto por várias fases distintas (fig. 2.2). Inicialmente um jogador na web escreve um conjunto de frases que compõem uma história. Cada uma dessas frases é processada e são extraídos os substantivos, dos quais alguns são escolhidos para ilustrar a narrativa.

Seguidamente cada um dos substantivos é enviado para os utilizadores na rua que devem tirar uma foto que seja representativa do nome recebido (ganhando com isso pontos). Quando esta é tirada, é automaticamente enviada para o servidor do jogo. Este por seu turno, re-encaminha a foto para dois utilizadores que são responsáveis pela sua validação. Finalmente quando uma foto é validada, esta e a respectiva palavra ficam automaticamente disponíveis para ilustrar qualquer história.

Eventualmente, quando todas as palavras seleccionadas de uma história sofrerem este processo, considera-se que esta está ilustrada, sendo posteriormente sujeita a uma avaliação por parte de um moderador humano. Finalmente as melhores narrativas são mostradas em Times Square.

Aspectos interessantes

Este trabalho apresenta algumas características interessantes, nomeadamente devido ao facto de promover a colaboração entre utilizadores que se encontram em contextos radicalmente diferentes (web e mundo real), tirando partido das características únicas de cada um.

A realização deste jogo expôs algumas diferenças extremamente marcantes entre estes dois mundos, nomeadamente no que toca à questão do tempo, isto é, uma espera de alguns minutos pode ser aceitável no mundo da web, mas no mundo real é completamente inaceitável.

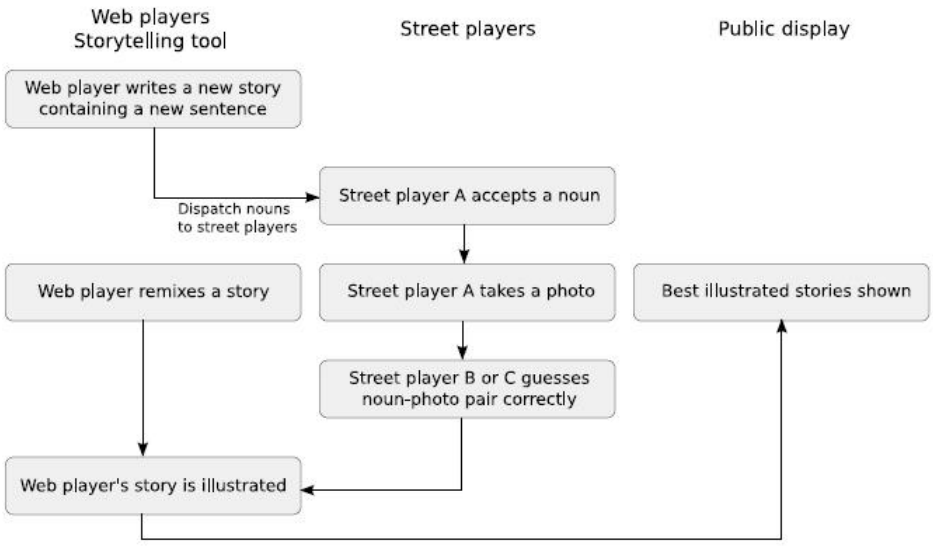

Figura 2.2: Fluxo do jogo Manhattan Story Mashups (de [TSN07])

2.2.3 Ubifit

O UbiFit [CKM+08] é um sistema que pretende auxiliar os utilizadores no sentido destes levarem um estilo de vida mais saudável. Este, permite que os utilizadores monitorizem a actividade física efectuada ao longo de uma semana, encorajando o cumprimento de objectivos específicos.

Arquitectura e funcionamento

O sistema UbiFit é composto por três componentes fundamentais: um dispositivo móvel dotado de um *glanceable display*, uma aplicação interactiva e um dispositivo (*fitness device*) [CBC+08] que permite inferir qual é a actividade física que o utilizador está a realizar num dado momento.

O *glanceable display* (fig. 2.3) usa uma representação estilizada das actividades físicas realizadas ao longo de uma semana, mostrando no ecrã do telemóvel, como *screensaver*, quais os objectivos que o utilizador já atingiu. Esta representação pode ser tão simples como por exemplo, uma imagem de um jardim, que no início da semana não tem vegetação, mas à medida que o utilizador vai fazendo exercício ou outras actividades saudáveis se vai enchendo com flores e outros elementos. A aplicação interactiva reside no dispositivo móvel e inclui informação detalhada sobre as actividades que foram realizadas e ainda um ”diário”onde é possível adicionar, editar ou remover actividades.

Finalmente, o *fitness device* é composto por um conjunto de sensores, em particular

Figura 2.3: *Glanceable display*(de [CKM+08])

um barómetro e um acelerómetro para inferir o tipo de actividades que o utilizador realiza, comunicando via bluetooth com o telemóvel várias vezes por segundo. A aplicação residente no telemóvel, tem a responsabilidade de agregar os dados recebidos de forma a obter uma visão mais abrangente sobre a actividade em curso.

Aspectos interessantes

Este trabalho apresenta uma aplicação que corre exclusivamente no dispositivo móvel, fazendo uso de alguns sensores externos. Um dos aspectos mais curiosos deste trabalho tem a ver com a utilização de um *glanceable display*, que permite que o utilizador se aperceba da informação que pretende ser veiculada de uma forma não intrusiva (no sentido em que pode ser simplesmente ignorada) e discreta.

2.2.4 Comparação entre diferentes sistemas

Com os sistemas estudados nesta categoria procurou dar-se uma visão abrangente das diferentes tipologias de aplicação que podem beneficiar de uma interacção entre um dispositivo móvel e os recursos fornecidos pela infra-estrutura.

Na prática verifica-se que existe um espectro de aplicações, onde num extremo se encontram aquelas que executam exclusivamente na infra-estrutura e no extremo oposto estão as aplicações que correm exclusivamente nos dispositivos móveis.

Entre estes dois pólos, podemos enquadrar o conjunto das aplicações cuja execução pode ser distribuída entre a infra-estrutura e equipamentos móveis.

No PIPE, define-se uma arquitectura genérica, orientada aos serviços, que permite a integração de diferentes serviços e aplicações. O objectivo é permitir acomodar as diversas tipologias de aplicação identificadas.

2.3 Plataformas de sistemas ubíquos para fornecimento de serviços

Nesta secção serão abordadas as diferentes plataformas ubíquas de fornecimento de serviços estudadas. Globalmente, estas soluções são bastante mais complexas, que os sistemas abordados na secção anterior, apresentando objectivos mais ambiciosos.

Nos sistemas abordados nesta secção observa-se uma divisão clara entre as abordagens baseadas em máquinas virtuais (e.g. [SF05, GC04]), e as restantes (e.g. [KBM+00, LJP+06, RRA08]), que geralmente consistem em arquitecturas orientadas aos serviços.

Nos sistemas que seguem a primeira abordagem, tipicamente é dada a possibilidade ao utilizador de executar as suas aplicações nos computadores presentes na infra-estrutura. No entanto estas aplicações não ficam integradas num ambiente de computação ubíqua que permita uma partilha e utilização mais racional dos recursos oferecidos.

Nos outros sistemas, tipicamente são utilizadas infra-estruturas orientadas aos serviços de forma a permitir à infra-estrutura disponibilizar um conjunto pré-definido de serviços e aplicações.

2.3.1 Cooltown

O projecto Cooltown [KBM+00], visa explorar as oportunidades que advêm do facto do mundo físico ser cada vez mais permeado por dispositivos computacionais, equipados com diferentes tecnologias de comunicação. O objectivo principal deste projecto consistiu em dotar pessoas, objectos e locais com uma presença web, providenciando diversos serviços e enriquecendo a experiência do utilizador.

Arquitectura e funcionamento

Em traços gerais, a abordagem principal deste sistema consiste em criar presenças web para os vários dispositivos e objectos que co-existem num espaço.

Um problema importante neste cenário, tem a ver com o facto de ser necessário detectar os URLs das páginas associadas aos objectos ou locais. Neste trabalho, foram contempladas duas abordagens de descoberta "física". Uma primeira designada por *direct sensing*, que recorre a transmissores associados aos objectos (e.g. infra-vermelhos), que emitem o URL respectivo. Estes por sua vez são capturados pelo dispositivo móvel do utilizador, sendo usados para mostrar a página web associada. Esta abordagem tem a vantagem óbvia de não requerer serviços adicionais para mostrar o conteúdo desejado.

Uma abordagem complementar, consiste no chamado *indirect sensing*, no qual o dispositivo do utilizador obtém um nome para o objecto, o qual pode ser usado como parâmetro de pesquisa num serviço de busca (cujo URL pode ser obtido usando *direct sensing*). A grande vantagem desta modalidade é permitir a atribuição de diferentes nomes, apropriados a diferentes contextos, que sejam mapeados no mesmo URL.

Outro aspecto relevante, é a necessidade de trocar informação com os vários objectos e dispositivos presentes no espaço (impressoras, projectores, etc). Estas trocas podem ser de dois grandes tipos: um primeiro, vocacionado para a transferência de conteúdos, usa um mecanismo designado por *eSquirt*, no qual é passado o URL do conteúdo a transferir para o receptor, acedendo este último aos dados desejados a partir do URL. O segundo tipo de troca de informação é vocacionada para o controlo do estado de um dispositivo, usando para tal formulários (HTML ou XForm).

Relativamente aos espaços, estes são geridos por um *place manager*, que é um servidor web que agrega informação relativa ao próprio local, dispositivos e utilizadores que nele se encontram.

Para permitir o contacto entre indivíduos, usa-se um serviço designado por Weblink, e retorna um URL adequado para que se possa efectuar a comunicação com a pessoa seleccionada. Este serviço guarda um link que vai sendo actualizado à medida que o utilizador se vai movendo. Numa situação em que não seja possível contactar o utilizador, o Weblink dá a possibilidade de se deixar uma mensagem, que será entregue quando este voltar a estar contactável.

Aspectos interessantes

O Cooltown é um sistema que visa criar uma "ponte"entre o mundo físico e o mundo da Web, associando informação e serviços a todo um conjunto de objectos, enriquecendo a experiência dos utilizadores.

Um aspecto interessante deste sistema, prende-se com o facto de usar a Web como base para construir um sistema desta natureza. Esta abordagem explora algumas particularidades importantes da web, em particular, o facto de ter baixas barreiras à entrada, uma interface minimalista e protocolos independentes dos serviços, linguagens ou dispositivos.

As aplicações que executam no PIPE podem exportar uma interface Web de forma a serem controladas a partir do cliente móvel. Uma das vantagens mais interessantes desta abordagem tem a ver com o facto da maioria dos equipamentos móveis já disporem de um browser, o que permite que este tipo de interface seja usado sem requisitos adicionais.

2.3.2 Celadon

O Celadon [NC05,LJP+06] é uma plataforma que pretende promover a colaboração entre dispositivos móveis e dispositivos presentes num determinado espaço (computadores, ecrãs, impressoras, etc), de forma a oferecer ao utilizador um conjunto de diferentes serviços. Esta abordagem visa explorar as vantagens que as várias classes de dispositivos oferecem. Por um lado, a mobilidade, personalização e crescente capacidade de armazenamento dos dispositivos móveis. Por outro, a largura de banda, poder de processamento e qualidade dos ecrãs que a infra-estrutura instalada no ambiente oferece.

Arquitectura e funcionamento

No modelo proposto pelo Celadon, os serviços disponibilizados encontram-se agregados em zonas, que correspondem a espaços concretos, como por exemplo: aeroportos, estações de comboio, centros comerciais, etc.

Cada uma destas zonas é controlada por um *Embedded Interaction Broker* (EIB), que é responsável por oferecer um conjunto de serviços aos dispositivos móveis sob a sua alçada. Uma entidade de mais alto nível, o *Zone Collaboration Broker*, é responsável por gerir um conjunto de zonas, podendo ainda oferecer alguns serviços que requeiram uma maior capacidade de memória ou processamento (fig. 2.4).

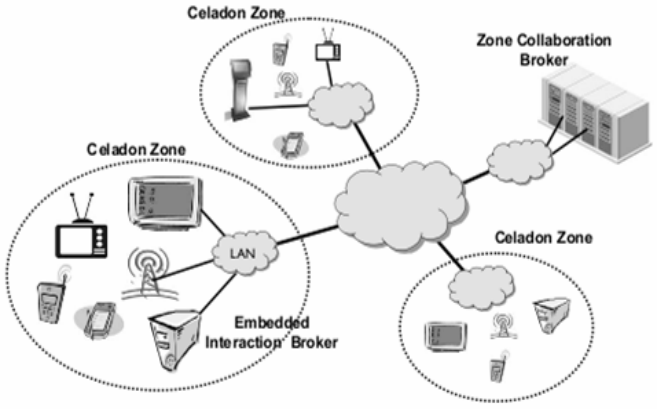

Figura 2.4: Arquitectura do sistema Celadon (de [LJP+06])

A descoberta do EIB, é feita através de uma query DNS após o dispositivo móvel se ter ligado à rede local da zona. A subsequente descoberta de serviços e comunicação é feita através de web-services.

Os serviços são disponibilizados aos utilizadores com base na informação contextual, preferências e permissões que estes exibem.

Consideremos um breve exemplo, num espaço comercial equipado com este sistema, há um serviço de pesquisa de restaurantes, que dá não só as características de um restaurante, mas também a sua localização num mapa da área. Assim, quando um utilizador pretende saber quais são os restaurantes disponíveis, basta que o seu dispositivo móvel contacte o EIB e escolha o serviço adequado. Seguidamente pode interrogar o sistema e obter a informação que pretende.

Aspectos interessantes

Este sistema é um exemplo de uma arquitectura genérica, orientada aos serviços, que funciona num ambiente com uma grande quantidade de dispositivos computacionais distintos. De forma a lidar com a heterogeneidade inerente a um cenário desta natureza, recorre a uma abordagem baseada em standards, neste caso concreto web-services, de maneira a promover a interoperabilidade entre vários sistemas.

O desenho do nosso sistema foi influenciado pelo Celadon, na medida em que também usamos uma solução baseada em aplicações que são disponibilizadas pelo sistema e que podem ser consumidas pelos utilizadores. No entanto, o PIPE também permite que estes usem a infra-estrutura para instalar as suas próprias aplicações.

2.3.3 Slingshot

A proliferação de *hotspots* wireless torna possível que usando dispositivos móveis, se corram aplicações com requisitos mais elevados, recorrendo a outras máquinas para executar as partes computacionalmente mais exigentes.

Isto permite que as aplicações, que tradicionalmente correm num único espaço, em casa ou no escritório, possam acompanhar os utilizadores à medida que estes se movimentam e se associam a diferentes *hotspots*. Na prática, esta ideia de transportar as computações para uma rede mais próxima do utilizador, permite uma redução substancial na latência relativamente a um modelo de execução remota, uma vez que as trocas de informação passam a ser feitas na mesma rede local e não através da Internet.

Para clarificar um pouco o objectivo deste sistema, imaginemos que o utilizador, dispondo apenas do seu telemóvel e de uma ligação à rede pretende correr uma aplicação de reconhecimento de voz. Como se trata de uma aplicação que requer algum poder computacional para executar, uma solução seria usar o dispositivo móvel para fazer apenas a captura do áudio e depois transferir os dados através da internet para um servidor, que estaria a cargo do seu processamento. Uma desvantagem óbvia desta abordagem tem a ver com a latência inerente à comunicação com o servidor. Usando o Slingshot, seria possível ao utilizador replicar numa máquina da rede local a que este

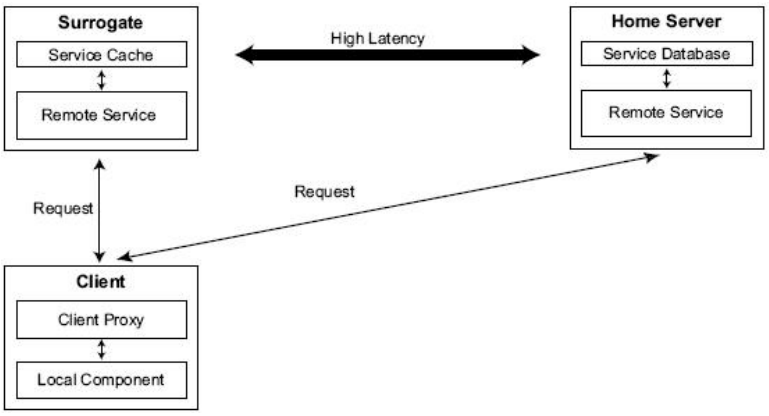

Figura 2.5: Arquitectura do sistema Slinghot (de [SF05])

está ligado, a aplicação que corre no servidor remoto, diminuindo assim o tempo de resposta.

Arquitectura e funcionamento

Em traços gerais, o Slingshot [SF05] usa uma estratégia de replicação de máquinas virtuais e aplicações, de forma a conseguir melhorar os tempos de resposta. Assim, cada *hotspot* tem um conjunto de computadores ligados, que estão disponíveis para correr réplicas (em máquinas virtuais distintas) das aplicações dos utilizadores. Essas réplicas são obtidas a partir de um servidor, bem conhecido pelo utilizador, que actua como réplica primária e é usado como repositório "confiável"do estado da aplicação.

A arquitectura do sistema é composta por três componentes: um servidor base do utilizador, o *home server*, um conjunto de máquinas disponíveis numa dada rede local, os *surrogates* e finalmente uma componente que corre no dispositivo móvel do utilizador, o cliente (fig. 2.5).

O *home server* actua como réplica primária das aplicações de um dado utilizador, correndo cada uma na sua própria máquina virtual e guardando não só o seu estado persistente, mas também o seu estado volátil.

Os *surrogates* instanciam as réplicas secundárias a partir do *home server*. Para tal, entra em contacto com esta entidade e obtém uma cópia do estado da aplicação. Uma vez terminada a transferência destes dados, é instanciada uma nova máquina virtual.

O cliente, pode aceder às réplicas através de um proxy, que é responsável pela descobertam instanciação, comunicação e destruição das réplicas.

Aspectos interessantes

O Slingshot mostra que esta aproximação permite melhorias de performance significativas, relativamente a um modelo de execução remota simples.

Adicionalmente este trabalho é bastante interessante, pois apresenta uma direcção um pouco diferente daquela de muitos outros trabalhos, na medida em que o cliente tem um papel mais activo na escolha das aplicações que recorrem à infra-estrutura existente.

O PIPE incorpora a capacidade do utilizador poder executar as suas próprias aplicações na infra-estrutura, sendo no entanto uma solução mais leve, baseada em código Java. Adicionalmente as novas aplicações ficam integradas no ambiente de computação que o sistema fornece, podendo fazer uso dos serviços fornecidos.

As soluções baseadas em máquinas virtuais apresentam alguns pontos positivos, relativamente às arquitecturas orientadas aos serviços como o PIPE. Por um lado, num modelo em que cada utilizador esteja a trabalhar na sua máquina virtual privada, é muito mais fácil limitar os recursos disponibilizados. Adicionalmente, é mais simples garantir o isolamento entre os diversos utilizadores. Por fim, uma solução desta natureza permite uma flexibilidade muito maior, na medida em que os utilizadores não estão limitados a executar aplicações numa linguagem específica (e.g. Java), tendo acesso a um conjunto de opções muito mais vasto.

2.3.4 AlfredO

O AlfredO é um sistema de *middleware* que faz a gestão dos serviços oferecidos num espaço ubíquo. Este permite ao utilizador interagir com os serviços fornecidos, usando o para tal seu dispositivo pessoal.

Esta plataforma apresenta uma arquitectura orientada aos serviços, que possibilita a distribuição, para os equipamentos dos utilizadores, do software que dos diferentes serviços fornecidos pelos dispositivos que compõem a infra-estrutura.

Arquitectura e funcionamento

Internamente, o AlfredO incorpora três mecanismos para atingir o nível de funcionalidade pretendido: um modelo de distribuição de software baseado em serviços, uma arquitectura multi-camada (multi tier) para os serviços e um modelo de apresentação independente do dispositivo.

Relativamente ao modelo de distribuição de software, este faz uso do R-OSGi [R-O09], que é baseado na plataforma OSGi[2]. Tipicamente a plataforma OSGi é utilizada

[2]http://www.osgi.org

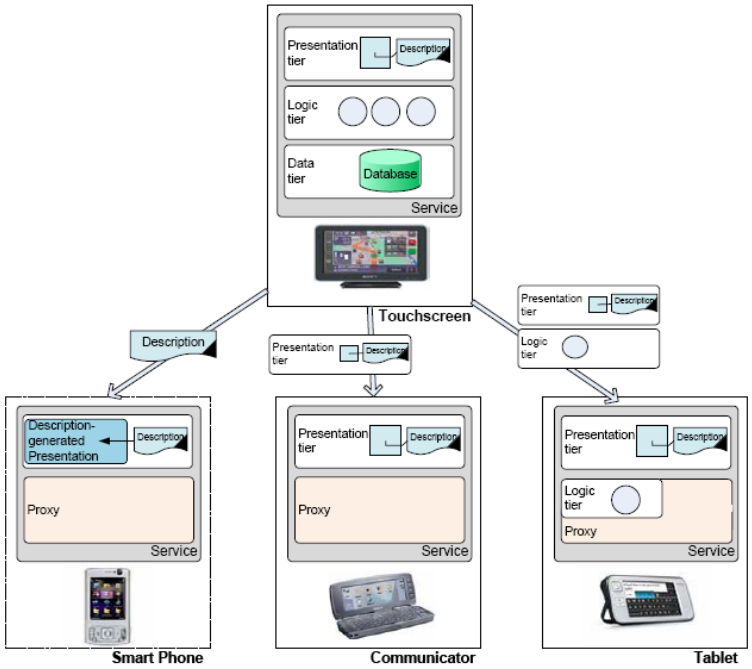

Figura 2.6: Arquitectura de um serviço (de [RRA08])

para decompor uma aplicação Java em módulos distintos, podendo o seu ciclo de vida ser gerido individualmente em tempo de execução.

Os serviços são divididos em três partes, seguindo uma estratégia em que cada serviço tem uma camada de apresentação, uma camada de lógica e uma camada de dados (fig. 2.6). A ideia subjacente é que estas três camadas possam ser distribuídas de maneira adequada ao equipamento utilizado e aos recursos disponíveis.

A pesquisa de serviços é tratada pelo R-OSGi. O cliente móvel pode obter do dispositivo de infra-estrutura dois elementos.

Primeiro, um proxy para o serviço, o que permite a sua posterior utilização. Segundo, a informação para gerar uma interface gráfica adequada ao dispositivo em causa. Esta pode fazer uso de bibliotecas como o AWT, ou caso não exista suporte é gerada uma interface HTML enriquecida com AJAX que é mostrada no browser do dispositivo.

No cenário de utilização mais comum, a proxy para o serviço é associada à interface gerada, sendo as operações efectuadas totalmente no dispositivo de infra-estrutura.

No entanto, também são suportados cenários mais complexos, em que são transferidos para o equipamento móvel não só os dados referidos anteriormente, mas também parte da lógica do serviço. Quando tal acontece, são instanciadas as proxies para os serviços listados no descritor do serviço, de forma a que estas possam ser utilizadas. O objectivo é reduzir a latência, melhorando o tempo de resposta da aplicação.

Aspectos interessantes

Este sistema vem mostrar a validade de uma abordagem baseada em serviços num cenário em que a maioria das interacções entre dispositivos assumem um carácter fortuito.

O AlfredO apresenta alguns aspectos muito interessantes, em particular a estratégia seguida na decomposição de um serviço, e a geração de uma interface gráfica independente do dispositivo.

Os resultados experimentais confirmam que se trata de um sistema eficiente, ocupando pouca memória no dispositivo móvel e oferecendo tempos de resposta interessantes.

O PIPE tem algumas semelhanças com o AlfredO, na medida em que também apresenta uma arquitectura orientada aos serviços que permite a disponibilização de um conjunto pré-definido de aplicações de sistema, cuja execução pode ser distribuída entre a infra-estrutura e o dispositivo móvel do utilizador. No entanto, o PIPE dá a possibilidade dos utilizadores executarem as suas próprias aplicações na infra-estrutura, o que lhe confere bastante flexibilidade.

2.3.5 Comparação entre diferentes sistemas

Os sistemas que fazem parte desta categoria serão comparados de acordo com quatro critérios: se requer aplicação no dispositivo móvel do utilizador; se requer algum registo prévio por parte do utilizador; se fornece um conjunto de serviços de infra-estrutura; se permite ao utilizador definir os seus próprios serviços.

A tabela 2.1, mostra o subconjunto dos sistemas estudados que pertence a esta categoria.

Sistema	Aplicação no disp. do cliente	Requer registo do utilizador	Serviços adicionais	
			Pré-definidos	Definidos pelo utilizador
Cooltown	Sim	Depende	Sim	Não
Celadon	Sim	Depende	Sim	Não
Slingshot	Sim	Não	Não	Sim
AlfredO	Sim	Não	Sim	Não

Tabela 2.1: Plataformas de sistemas ubíquos para fornecimento de serviços

Uma constatação interessante, prende-se com o facto da maioria destes sistemas não suportar a definição de aplicações ou serviços por parte do utilizador.

2.4 Considerações finais

Nas secções anteriores foram apresentados alguns sistemas que reflectem o trabalho efectuado nesta área, e que apresentam diferentes abordagens à problemática da combinação de dispositivos móveis com os recursos fornecidos pela infra-estrutura de um espaço inteligente.

Começamos por apresentar exemplos de aplicações que fazem uso, em maior ou menor grau, dos recursos disponibilizados pela infra-estrutura. Estas cobrem as diferentes tipologias de aplicação que podem existir neste cenário.

Relativamente às plataformas ubíquas de fornecimento de serviços, constata-se que à excepção daquelas baseadas em máquinas virtuais, as restantes não permitem ao utilizador efectuar a execução de aplicações definidas pelo utilizador.

O sistema PIPE, combina características dos diferentes sistemas abordados neste capítulo, de forma a acomodar diferentes tipologias de aplicação identificadas anteriormente. Este apresenta uma arquitectura orientada aos serviços, na qual as aplicações podem fazer uso dos diferentes recursos disponibilizados pela infra-estrutura. Aplicações essas, que tanto podem pertencer à infra-estrutura, como ser carregadas pelo utilizador, oferecendo uma maior flexibilidade.

3

Arquitectura

Neste capítulo apresenta-se a arquitectura do sistema PIPE. Este pretende criar uma plataforma computacional ubíqua que permita a interacção entre dispositivos da infra-estrutura e dispositivos móveis.

Este capítulo divide-se em duas partes. Na primeira é feita uma apresentação de carácter mais geral do sistema, explicitando quais são as suas principais componentes e como estas interagem de forma a alcançar os objectivos propostos. Na segunda, analisam-se detalhadamente cada uma das componentes, e termina-se com uma apresentação do modelo de segurança subjacente.

3.1 Arquitectura do sistema

O sistema PIPE exibe uma arquitectura orientada aos serviços e é formado por dois tipos de componentes: os computadores e sensores associados (e.g. câmaras) que se encontram espalhados pelo espaço físico, designados por Infrastructure Nodes, e os dispositivos móveis dos utilizadores que frequentam esse espaço.

A figura 3.1, apresenta um exemplo da instalação deste sistema num espaço público (e.g. centro comercial). Nesta é possível observar a interacção entre os dispositivos móveis dos utilizadores e os equipamentos presentes na infra-estrutura. As interacções entre esses dispositivos podem ocorrer através de diferentes tecnologias de comunicação, por exemplo bluetooth ou Wi-Fi.

Estes equipamentos móveis podem executar a aplicação do sistema, permitindo aceder às funcionalidades mais avançadas que o PIPE oferece, como por exemplo ex-

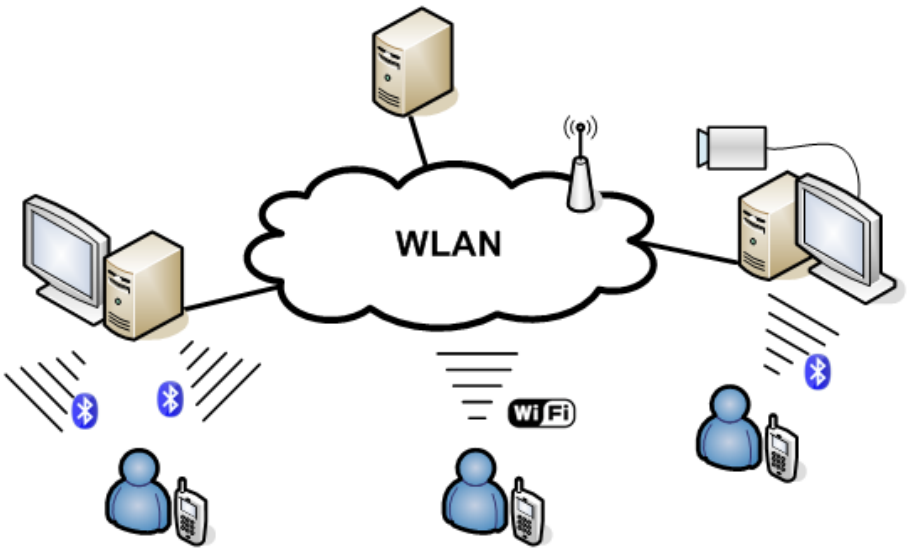

Figura 3.1: Cenário geral do sistema.

ecutar parcial ou totalmente aplicações definidas pelo utilizador nos Infrastructure Nodes. Caso esta aplicação móvel não esteja presente, as aplicações apenas executam nos nós da infra-estrutura. Um exemplo deste tipo de aplicações é a aplicação de publicidade desenvolvida e que será abordada em maior detalhe no capítulo 5.2.1. Esta executa totalmente num Infrastructure Node, detectando os dispositivos bluetooth presentes na vizinhança, não requerendo interacção explícita com o utilizador ou o seu equipamento pessoal. Com base nas deslocações destes dispositivos é possível inferir quais os interesses dos utilizadores e mostrar publicidade que vá de encontro a esses interesses.

Quando um utilizador tem a aplicação móvel instalada no seu dispositivo pessoal, pode utilizá-la para se conectar a um Infrastructure Node presente. Este possui à partida um conjunto de aplicações que podem ser usadas imediatamente, por exemplo num centro comercial uma aplicação que permita fazer pesquisas de produtos em diferentes lojas, mostrando os resultados obtidos num ecrã público ou no próprio dispositivo móvel.

Para garantir uma maior flexibilidade também é possível ao utilizador correr na infra-estrutura, total ou parcialmente, as suas próprias aplicações. Uma aplicação que se enquadra neste contexto é a aplicação de informação estendida sobre produtos que desenvolvida. Esta aplicação permite que o utilizador, obtenha informação sobre um produto a partir de serviços web, com base no seu código de barras. Neste caso a componente que contacta os diferentes serviços pode correr na infra-estrutura, tirando partido do maior poder computacional e largura de banda

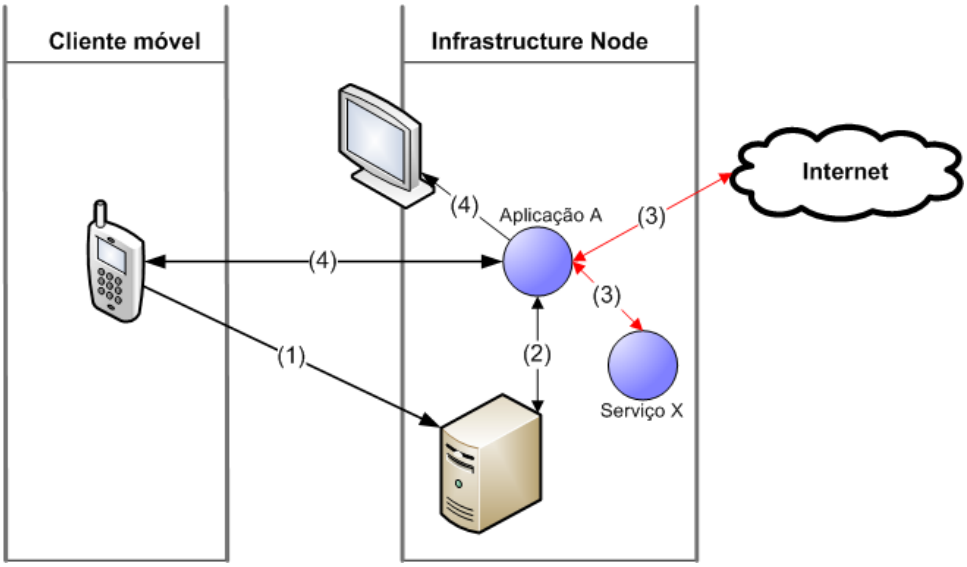

Figura 3.2: Interacção global entre as várias entidades do sistema.

Em traços gerais, o cenário descrito anteriormente e as interacções entre estas duas grandes entidades podem ser sumariadas através da figura 3.2.

Inicialmente o cliente móvel contacta o Infrastructure Node, emparelhando-se com o mesmo e obtendo informação sobre as aplicações disponíveis (1). Num passo seguinte, o utilizador escolhe a aplicação que pretende executar, que pode ser local ao nó ou externa. Neste último caso, a aplicação pode ser transferida a partir do próprio dispositivo móvel ou de um servidor na rede. Esta aplicação, independentemente de já existirem instâncias a correr, é executada num contexto protegido, que garante o seu isolamento das restantes aplicações e serviços que se encontrem a correr naquele momento. A aplicação pode exportar uma interface web que permite o seu controlo a partir do cliente móvel (2). Esta aplicação pode aceder a diferentes serviços disponibilizados pelos vários nós da infra-estrutura (e.g. serviço de câmaras, serviço de detecção bluetooth) ou, se tiver permissões para tal, aceder a recursos na Internet (3). Tipicamente, após os diferentes serviços serem consultados, os resultados podem ser mostrados num ecrã público associado ao Infrastructure Node, se algum, ou enviados para a aplicação que corre no dispositivo móvel do utilizador (4). Este processo poderá variar dependendo um pouco das especificidades de cada aplicação, podendo os passos 3 e 4 ser repetidos para diferentes pedidos do cliente.

No PIPE são suportadas duas formas de permitir o acesso a uma aplicação que execute na infra-estrutura a partir do cliente móvel. A primeira, já referida acima, consiste na capacidade da aplicação exportar uma interface web que pode ser acedida usando o browser do próprio dispositivo móvel. A segunda, consiste em associar à aplicação

que executa na infra-estrutura, uma aplicação móvel que pode ser descarregada e iniciada automaticamente no cliente móvel.

No cenário descrito anteriormente, ambas as componentes desempenham um papel. Porém, a infra-estrutura está preparada para permitir outros cenários de utilização. Por um lado podem existir aplicações que correm na infra-estrutura e que não requerem uma interacção explícita com o dispositivo móvel do utilizador, tornando por isso a componente móvel opcional. Por outro lado também podem existir aplicações que correm totalmente nos dispositivos móveis dos utilizadores, acedendo directamente aos serviços fornecidos pela infra-estrutura.

3.1.1 Conceitos: Aplicação e serviço

No contexto deste sistema os componentes de software que correm sob o seu controlo podem dividir-se em aplicações e serviços.

Um serviço é uma componente de software que fornece uma funcionalidade específica a outros serviços ou aplicações (e.g. serviço de detecção bluetooth, serviço de câmaras etc.).

Uma aplicação é uma componente de software, que pode correr total ou parcialmente na infra-estrutura e oferece um conjunto de funcionalidades ao utilizador final. Cada aplicação está associada a um único utilizador, podendo aplicações com múltiplos utilizadores ser suportadas através da utilização de serviços que forneçam um meio de comunicação entre elas.

3.2 Componentes do sistema

3.2.1 Infrastructure node

O Infrastructure Node é o componente fundamental deste sistema, tendo por missão oferecer um suporte computacional onde diferentes aplicações podem ser executadas. Estas podem correr autonomamente, ou ser controladas pelos dispositivos móveis dos utilizadores (assumindo que estes possuem a aplicação cliente instalada). De forma a facilitar o desenvolvimento de novas aplicações e também para potenciar a reutilização dos vários recursos que a infra-estrutura oferece, os Infrastructure Nodes podem ainda fornecer serviços. Estes podem ser consumidos por diferentes aplicações, ou compostos em serviços de mais alto nível.

A estrutura interna do Infrastructure Node, apresentada na figura 3.3 pode-se subdividir em quatro módulos: o módulo de comunicação, que é responsável pelas comunicações com os clientes móveis. O gestor de aplicações, que tem por missão administrar as

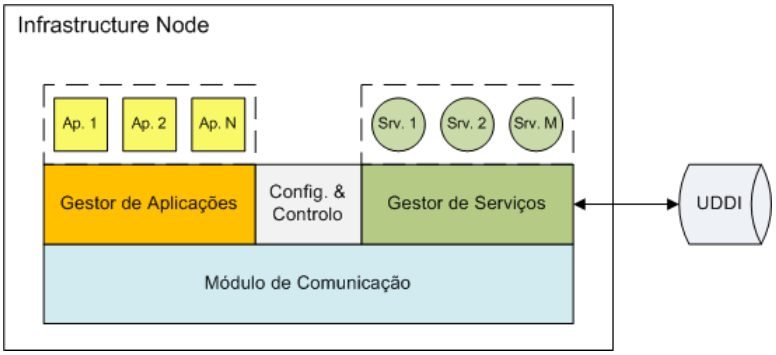

Figura 3.3: Arquitectura do infrastructure node.

aplicações que executam na infra-estrutura. O gestor de serviços que controla os serviços que correm localmente no Infrastructure Node, e o módulo de configuração que permite definir diversos aspectos relacionados com a configuração do Infrastructure Node, incluindo as aplicações e serviços disponíveis inicialmente.

De forma a acomodar uma maior variedade de dispositivos, o gestor de comunicações suporta a utilização de diferentes tecnologias de comunicação, em particular bluetooth e tecnologias de comunicação baseadas em IP (e.g. Wi-Fi, GPRS etc.).

O gestor de aplicações é responsável por fazer toda a gestão das aplicações que executam no Infrastructure Node, quer se tratem de aplicações residentes na infra-estrutura quer sejam aplicações transferidas pelos utilizadores. Todas as aplicações e serviços executam num contexto protegido, havendo a possibilidade de associar a cada diferentes permissões de segurança.

Um aspecto importante prende-se com as diferentes formas como o utilizador pode interagir com uma aplicação que esteja a correr na infra-estrutura a partir do seu dispositivo móvel. Actualmente são suportadas duas opções: a própria aplicação pode exportar uma interface web que é colocada online quando esta é iniciada, ou então pode ser definida uma aplicação móvel associada que é instalada e lançada no dispositivo móvel.

A primeira opção, baseada em interfaces web, é adequada para aplicações em que o método de controlo não é muito complexo. A grande vantagem desta abordagem tem a ver com o facto de apenas ser necessário que o equipamento móvel disponha de um browser e conectividade IP (e.g. Wi-Fi), sendo estes requisitos cumpridos por um número crescente de dispositivos. Adicionalmente, esta modalidade permite que rapidamente se desenvolvam interfaces de controlo, sem entrar nas especificidades que caracterizam actualmente o panorama do desenvolvimento para dispositivos móveis e tirando partido das várias ferramentas que já existem no âmbito do desenvolvimento web.

No protótipo implementado, são utilizadas aplicações web Java que são instaladas num servidor a executar localmente.

A segunda abordagem, baseada na utilização de uma aplicação a instalar automaticamente no cliente, vem colmatar algumas das limitações ao meio de controlo descrito anteriormente. Esta abordagem é adequada para aplicações mais complexas, permitindo uma adaptação muito melhor às especificidades da aplicação em causa, no entanto obriga a criar uma aplicação específica para dispositivos móveis.

Como já foi referido, as diversas aplicações que correm no Infrastructure Node podem fazer uso dos diferentes serviços fornecidos (internamente ou por outros nós). Estes são geridos pelo gestor de serviços, que é responsável pelo controlo dos serviços que correm localmente no nó, bem como pelo seu registo no directório global de serviços. Para facilitar a criação e utilização dos serviços, é importante que estes sejam o mais standard possível. Neste contexto, decidiu-se que os serviços seriam definidos como web-services SOAP [SOA09]. Tendo em conta este facto, a utilização de um servidor de registo UDDI [UDD09] para manter a informação dos serviços existentes no sistema tornou-se uma escolha natural.

Quando uma aplicação requer um serviço, esta contacta o gestor de serviços, que deve verificar se o serviço requerido existe. Caso exista, é disponibilizada a informação que permite ao código do cliente (que está embutido na aplicação) aceder ao serviço em causa. Adicionalmente, aquando de um pedido de um serviço a aplicação pode especificar algumas preferências, em particular se o serviço requerido deverá ser local ao nó onde esta corre ou não. Tal como já foi referido, também é possível aos serviços utilizarem outros serviços, de forma a fornecer às aplicações operações de mais alto nível.

Quando tomadas no seu conjunto, as funcionalidades oferecidas pelos Infrastructure Nodes apresentam alguma complexidade. De forma a facilitar a gestão interna a cada nó, ortogonalmente às componentes que descrevemos anteriormente, existe um módulo de configuração. Este, não só controla alguns aspectos de natureza mais operacional da plataforma (portas utilizadas, localização do repositório de serviços, etc.) mas também permite definir de forma simples quais são as aplicações e serviços locais que estão disponíveis para executar.

3.2.2 Cliente móvel

O cliente móvel é a segunda grande componente sistema desenvolvido. Esta consiste no dispositivo móvel do utilizador, no qual pode executar a aplicação móvel do sistema PIPE. Esta, apesar de não ser absolutamente obrigatória, permite tirar partido das funcionalidades mais avançadas que o sistema oferece. Na prática, o cliente móvel

equipado com a esta aplicação, comporta-se como um gestor das aplicações que o utilizador pode usar no contexto deste sistema. Estas aplicações podem ser de dois tipos distintos. O primeiro tipo, consiste nas aplicações que correm na infra-estrutura nativamente e às quais o cliente pode aceder. O segundo tipo é composto pelas aplicações que o utilizador possui e cuja execução pode ser solicitada à infra-estrutura. Neste caso, as aplicações podem executar total, ou parcialmente num Infrastructure Node, sendo o seu código transferido a partir do dispositivo móvel ou a partir de um servidor remoto.

Após o lançamento de uma aplicação num dos nós, o cliente móvel é ainda responsável pela inicialização do meio de controlo que esta exporta. Como já foi referido, no protótipo actual são suportadas duas possibilidades distintas. A primeira, consiste numa interface web exportada pela aplicação, sendo apenas necessário iniciar o browser do dispositivo móvel no endereço apropriado. A segunda consiste numa componente de software a executar no cliente, caso em que é necessário instalar e iniciar essa aplicação móvel.

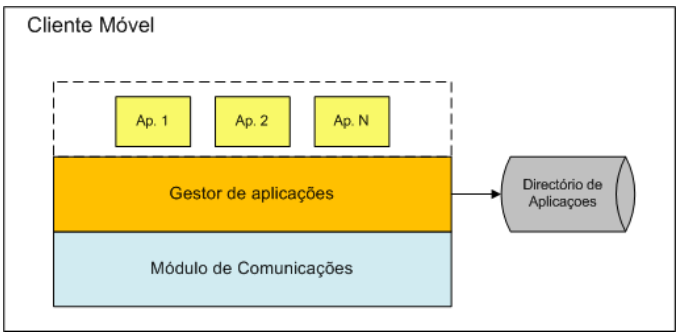

Figura 3.4: Arquitectura do cliente móvel.

Internamente, o cliente móvel possui uma estrutura um pouco semelhante ao Infrastructure Node. Este é composto por dois módulos distintos (figura 3.4): um primeiro módulo, cuja missão é gerir o processo de conexão do dispositivo móvel com a infra-estrutura e todas as subsequentes comunicações. De forma a garantir uma maior flexibilidade, quer em termos da quantidade de dispositivos móveis suportados, quer em termos das próprias interacções com a infra-estrutura, este módulo suporta várias tecnologias de comunicação (IP e bluetooth, no protótipo actual).

O segundo módulo consiste num gestor de aplicações, que concentra em si duas funções principais. Por um lado, é responsável por manter um directório das aplicações do utilizador, as quais podem ser enviadas para correr na infra-estrutura. Por outro lado é responsável por contactar a infra-estrutura e dar a opção ao utilizador de executar as aplicações que são oferecidas pelo Infrastructure Node.

3.3 Modelo de segurança

No contexto de um sistema como o PIPE, a questão da segurança é bastante premente. Num cenário onde podem coexistir diferentes serviços e aplicações potencialmente de diferentes utilizadores, podemos identificar essencialmente dois problemas: a protecção da infra-estrutura face à execução de software arbitrário e o controlo de acessos de uma aplicação relativamente aos serviços que pode utilizar.

Relativamente ao primeiro problema, é importante por um lado garantir que se atribuem diferentes permissões consoante se trate de uma aplicação ou serviço de infra-estrutura ou de um utilizador, podendo ainda variar consoante se trate de um utilizador autenticado ou não. Por outro lado, cada aplicação (ou serviço) deve executar num contexto protegido, isolado dos restantes.

Relativamente à questão do controlo de acessos de uma aplicação relativamente aos serviços que pode utilizar, uma solução possível poderia passar por combinar a utilização do sistema *Kerberos* com um sistema de *capacidades*. Neste modelo, um cliente podia apresentar aos Infrastructure Nodes um *ticket* com as suas permissões.

Naturalmente que este modelo de segurança responde apenas de forma básica às problemáticas deste tópico, carecendo de algum desenvolvimento até que se chegue a uma solução completa, passível de ser implementada num cenário real.

Adicionalmente, seria interessante providenciar mecanismos que permitam certificar ou auditar os resultados das computações efectuadas na infra-estrutura. O objectivo é evitar ou identificar utilizações ilícitas dos recursos disponibilizados (e.g download de conteúdos ilegais), eventualmente procedendo à responsabilização dos utilizadores envolvidos.

4

Implementação

Neste capítulo descreve-se detalhadamente a implementação do protótipo do sistema PIPE. Este segue o desenho apresentado no capítulo 3, abordando detalhadamente cada um dos módulos que foram introduzidos anteriormente.

4.1 Aplicações e serviços

No nosso protótipo tanto as aplicações como os serviços são programas Java, que se definem como tal através da conjugação de dois elementos distintos: por um lado pelo conjunto de interfaces que implementam e por outro através de um ficheiro de configuração XML associado.

4.1.1 Aplicações

As aplicações que executam na infra-estrutura devem implementar a interface *InfrastructureApp*. Adicionalmente, a classe que implementa esta interface deve apresentar um construtor sem argumentos. A interface *InfrastructureApp*, apresentada na listagem 4.1, permite ao sistema executar e interagir de forma genérica com uma aplicação que executa na infra-estrutura.

Através do método *setServiceManager* é possível dar à aplicação conhecimento do gestor de serviços do Infrastructure Node, o que permite que esta utilize os serviços já existentes no sistema ou que instale novos serviços. Para efectuar a instalação de novos serviços, esta recebe um conjunto de objectos do tipo *ServiceLocator*. Cada qual possui uma lista de localizações, obtidas com base no ficheiro de configuração da aplicação, e

a partir das quais é possível descarregar o código e o ficheiro de configuração associado ao serviço.

Listagem 4.1: Interface InfrastructureApp

```
1 public interface InfrastructureApp {
2     public void setServiceManager(BasicServiceManager serviceManager,
          Map<String, ServiceLocator> services);
3     public void init(Properties props);
4     public void startApp();
5     public void stopApp();
6 }
```

O método *init* permite fornecer à aplicação um conjunto de parâmetros de configuração baseados na informação contida no ficheiro XML de configuração da aplicação.

O método *startApp*, é o método responsável pela inicialização da aplicação após esta receber a informação de configuração e a referência para o gestor de serviços.

O método *stopApp* é utilizado para terminar a aplicação, solicitando a libertação de quaisquer recursos que entretanto tenham sido adquiridos.

O gestor de serviços da infra-estrutura, a que as aplicações têm acesso, implementa a interface *BasicServiceManager*, apresentada na listagem 4.2. As funcionalidades oferecidas por esta interface podem ser utilizadas para que a aplicação possa fazer uso dos serviços fornecidos pelo sistema. Como se pode constatar, esta apresenta duas operações distintas:

A primeira (*getServiceBindingUrl*) é utilizada para obter o URL do WSDL [wsd09], que permite que a aplicação faça uso desse serviço. Adicionalmente é possível especificar mais algumas propriedades, que indicam por exemplo, se o serviço deve ser local ao Infrastructure Node onde a aplicação corre.

Quando a aplicação requer um serviço que não está disponível no sistema, esta pode efectuar a sua instalação (método *installService*). Neste contexto, tipicamente usa-se a informação presente num dos objectos do tipo *ServiceLocator* que são passados à aplicação.

Listagem 4.2: Interface BasicServiceManager

```
1 public interface BasicServiceManager {
2     String getServiceBindingUrl(String name, ServicePrefs prefs);
3     public void installService(String name, String jarFile, String confFile
          , String serviceLocation);
4 }
```

Relativamente ao ficheiro de configuração da aplicação (anexo A.1), este define as seguintes propriedades fundamentais para permitir a sua inicialização: nome da

aplicação (que será usado internamente como identificador único); o URL do ficheiro jar (ou war) da aplicação; o nome da classe que implementa a interface *InfrastructureApp*, e que será usada como ponto de entrada; a informação sobre quais as interfaces exportadas - Web ou Aplicação Java2 ME; a informação sobre os serviços requeridos. Para além disso, é possível definir um ficheiro de configuração com uma sintaxe específica à aplicação em causa.

4.1.2 Serviços

Um serviço para correr num Infrastructure Node deve implementar a interface *InfrastructureService*, apresentada na listagem 4.3. Esta interface inclui um conjunto de métodos, que permitem fazer a gestão do ciclo de vida de um serviço, incluindo a sua inicialização e configuração. Adicionalmente, e de forma análoga às aplicações também é possível definir um ficheiro de configuração específico ao serviço.

O método *init* é utilizado para passar ao serviço um conjunto de informações de configuração. Entre estas, pode-se incluir a porta à qual este deve estar associado, ou algum ficheiro de configuração específico que esteja definido no ficheiro XML de configuração.

Os métodos *startService* e *stopService* são utilizados para fazer a gestão do ciclo de vida do serviço, sendo o primeiro usado para inicializar o serviço e o segundo usado para terminar o serviço, eventualmente libertando os recursos que entretanto tenham sido adquiridos.

Para fornecer informações adicionais relativamente ao serviço em causa, esta interface oferece os métodos: *getServiceName*, que obtém o nome do serviço; e *getWsdlUrl* que é obtém o URL do WSDL associado ao serviço.

Para suportar um cenário onde os serviços podem ser compostos em serviços de mais alto nível, de forma análoga às aplicações, existe o método *setServiceManager*, que permite que faça uso de outros serviços já existentes, ou instale novos serviços.

Listagem 4.3: Interface InfrastructureService

```
1 public interface InfrastructureService {
2     public void init(Properties props);
3     public void startService();
4     public void stopService();
5     public String getWsdlUrl();
6     public String getServiceName();
7     public void setServiceManager(BasicServiceManager serviceManager, Map<
          String, ServiceLocator> services);
8 }
```

Os ficheiros de configuração associados a cada serviço definem o seguinte conjunto de propriedades: o nome do serviço (que será usado internamente pelo infrastructure node); o ficheiro jar com o respectivo código; o ponto de entrada no serviço, isto é, a classe que implementa a interface *InfrastructureService*; e opcionalmente um ficheiro de configuração específico ao serviço.

4.2 Infrastructure node

4.2.1 Módulo de comunicação

O módulo de comunicação tem como missão fazer a gestão das comunicações entre o Infrastructure Node e o cliente móvel, abstraindo as outras componentes do sistema dos detalhes dessa tarefa. No protótipo actual são suportadas diferentes tecnologias de comunicação (bluetooth e IP), no sentido de permitir acomodar um conjunto mais vasto de equipamentos.

O módulo de comunicação implementa um serviço de emparelhamento, que funciona sobre bluetooth e fornece os endereços dos restantes serviços ao cliente móvel.

Na prática, o endereço deste serviço é o único endereço que o cliente móvel necessita de conhecer para poder aceder às funcionalidades do Infrastructure Node. À medida que os restantes servidores, internos ao nó, vão sendo iniciados, os respectivos endereços são dados a conhecer a este serviço. Quando o serviço de emparelhamento obtém todos os endereços (bluetooth e IP em ambas as funcionalidades), é efectuada a sua inicialização passando a disponibilizar essa informação aos clientes móveis.

Este módulo fornece o suporte para todo o processo de instalação de novas aplicações ou serviços e inicialização de aplicações já existentes na infra-estrutura. Para tal, o módulo de comunicação executa a troca de mensagens entre diferentes entidades. De seguida apresentam-se brevemente os protocolos de comunicação executados pelo sistema.

Protocolo de instalação de novas aplicações

Do ponto de vista do módulo de comunicações, a instalação de uma nova aplicação na infra-estrutura decorre em três passos distintos.

Inicialmente, o cliente móvel envia para o Infrastructure Node uma mensagem do tipo *ConfigMessage*, onde consta o ficheiro de configuração XML da aplicação a instalar. Seguidamente, este documento é processado, obtendo-se a localização do arquivo jar (ou war) com o código da aplicação. Este código pode residir no próprio dispositivo móvel, caso em que é enviada uma mensagem do tipo *FileRequestMessage*, onde consta

o nome do ficheiro a obter. O cliente responde com uma mensagem do tipo *FileMessage*, na qual consta o ficheiro solicitado. Na situação em que o arquivo com o código da aplicação reside num servidor remoto, este pode ser obtido através de HTTP.

No final deste processo, o gestor de aplicações tem todos os elementos que necessita para proceder à inicialização da aplicação, como veremos adiante.

Protocolo para execução de aplicações pré-definidas

A inicialização de uma aplicação existente na infra-estrutura obedece a um protocolo mais simples. Num primeiro passo, o cliente móvel envia uma mensagem do tipo *AppListMessage* que solicita a lista das aplicações passíveis de ser iniciadas pelo utilizador. Esta lista vem no formato de um documento XML, que é transferido para o cliente numa mensagem do tipo *FileMessage*.

Após o utilizador efectuar a escolha de uma das aplicações, é enviada uma mensagem do tipo *AppInitMessage*, onde consta o identificador único da aplicação a iniciar, que na nossa implementação consiste apenas no respectivo nome.

Inicialização da interface no dispositivo móvel

Independentemente de se tratar de uma aplicação já existente, ou de uma nova aplicação instalada, esta pode oferecer um de dois meios de interacção com o cliente móvel.

Caso se trate de uma aplicação que exporte uma interface Web, após esta ser colocada a executar pelo gestor de aplicações, é enviada uma mensagem do tipo *InterfaceMessage*, com o URL que permite ao cliente móvel abrir o seu browser no endereço adequado.

Na situação em que a aplicação requer uma componente que corra no dispositivo móvel, é enviada para a aplicação do sistema no dispositivo móvel, uma mensagem do tipo *MidletMessage*. Esta mensagem inclui a informação que permite efectuar o download dessa aplicação móvel e a sua posterior inicialização.

4.2.2 Gestor de aplicações

O gestor de aplicações é o módulo que tem como objectivo fazer a gestão das aplicações que executam num Infrastructure Node. Este assenta fundamentalmente na interacção entre os quatro componentes apresentados na figura 4.1.

O *AppManager* é responsável por interagir directamente com o módulo de comunicação, encaminhando os dados recebidos para as restantes componentes que compõem o gestor de aplicações.

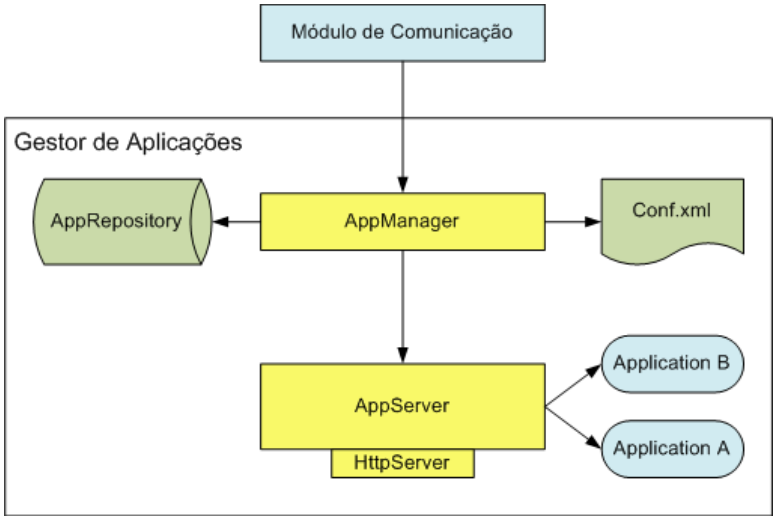

Figura 4.1: Estrutura interna do gestor de aplicações.

O *AppRepository* tem por missão manter a informação das aplicações de infra-estrutura que estão contempladas no ficheiro de configuração do Infrastructure Node.

O processo de inicialização de uma aplicação (em qualquer das modalidades) é tratado ao nível do *AppServer*. Este recebe os ficheiros de configuração XML e os arquivos jar (ou war) com o código da aplicação em causa, e utiliza estes dois elementos para efectuar a respectiva inicialização.

Adicionalmente, para suportar as interfaces de controlo Web que podem estar associadas às aplicações, o *AppServer* dispõe de um servidor HTTP interno.

O processo de inicialização de uma aplicação tem em conta os parâmetros presentes no ficheiro XML de configuração [1] e tem como objectivo iniciar um contexto protegido e devidamente configurado onde a aplicação possa correr, usufruindo dos serviços que a infra-estrutura proporciona.

Do ponto de vista operacional, a inicialização de uma aplicação desenrola-se em três fases distintas. Inicialmente, o arquivo com o código da aplicação é processado, sendo os seus conteúdos extraídos para uma directoria temporária que fica associada à aplicação. De seguida é criado um *classloader* específico para esta aplicação, que será usado por seu turno para estabelecer o contexto isolado onde a esta irá executar. Associado às classes que compõem a aplicação, define-se o conjunto de permissões de segurança das mesmas.

De seguida, é instanciado um objecto da classe indicada no ficheiro de configuração como ponto de entrada da aplicação, usando o construtor por omissão. Usando os

[1] Os parâmetros de configuração são: nome da aplicação, nome do ficheiro de configuração específico (se existir), qual o ponto de entrada, os serviços dos quais depende e se exporta alguma interface de controlo Web ou Java 2 ME.

métodos definidos na interface *InfrastructureApp*, procede-se à inicialização e configuração da aplicação.

Finalmente, é verificado se a aplicação exporta alguma interface Web, ou se utiliza uma aplicação Java 2 ME para efectuar o seu controlo a partir do cliente. No fim é invocado o método *startApp* da aplicação, que sinaliza que esta já se encontra configurada e que pode iniciar a sua execução, fazendo uso dos serviços do sistema.

Relativamente aos meios de controlo exportados, como já foi referido estes podem ser de dois tipos, aplicações Web ou aplicações Java 2 ME.

Quanto ao primeiro tipo, o *AppServer* quando é iniciado lança um servidor HTTP. Na nossa implementação, utilizámos para o efeito o servidor Jetty[2], que permite não só utilizar conteúdos Web estáticos (e.g. páginas HTML), mas também suporta aplicações Web Java com conteúdos dinâmicos através de tecnologias como JSP ou Java Servlets.

A interface Web deve estar incluída no arquivo da aplicação, ficando disponível na directoria temporária associada à aplicação, quando esta é instalada. Essa directoria é posteriormente associada ao servidor Web, sendo usada para servir os diversos conteúdos presentes. Para associar esta interface Web à aplicação instalada, é necessário fornecer ao Jetty a directoria da aplicação web. Adicionalmente é ainda necessário criar um contexto que permita à componente web aceder aos serviços e funcionalidades oferecidas pelo Infrastructure Node. Quando este processo é concluído, o URL da interface Web (e.g. http://10.171.133.201:9090/web/8c9pd6/index.jsp) pode ser disponibilizado ao cliente móvel.

No segundo caso, em que a aplicação é controlada a partir do cliente através de uma aplicação móvel, é apenas feito o registo do URL onde é possível obter a mesma e é da responsabilidade do cliente fazer o pedido de transferência para subsequente instalação.

Seguidamente será abordado com detalhe o processo de execução de aplicações de infra-estrutura e de instalação de aplicações definidas pelo utilizador.

Execução de aplicações de Infra-estrutura

A execução de aplicações na infra-estrutura inicia-se após o utilizador escolher a aplicação que pretende correr a partir da listagem obtida pelo cliente móvel. A figura 4.2 apresenta uma visão esquematizada deste processo. Inicialmente, é enviado para a infra-estrutura o identificador da aplicação seleccionada (1), sendo encaminhado para o *AppManager*, que é responsável por verificar se tal aplicação existe no repositório de aplicações de infra-estrutura (2). Se tal acontecer, obtém o ficheiro de configuração associado, tipicamente a partir do disco rígido (3). Este ficheiro é validado (4) e é obtido

[2]http://www.mortbay.org/jetty/

o arquivo jar onde se encontra o código da aplicação (5). Finalmente, tendo estes elementos é possível iniciar a aplicação (6).

Cada instância de uma aplicação está associada a um único utilizador. Para suportar aplicações onde existam múltiplos utilizadores é possível utilizar serviços que forneçam o suporte à comunicação entre várias aplicações.

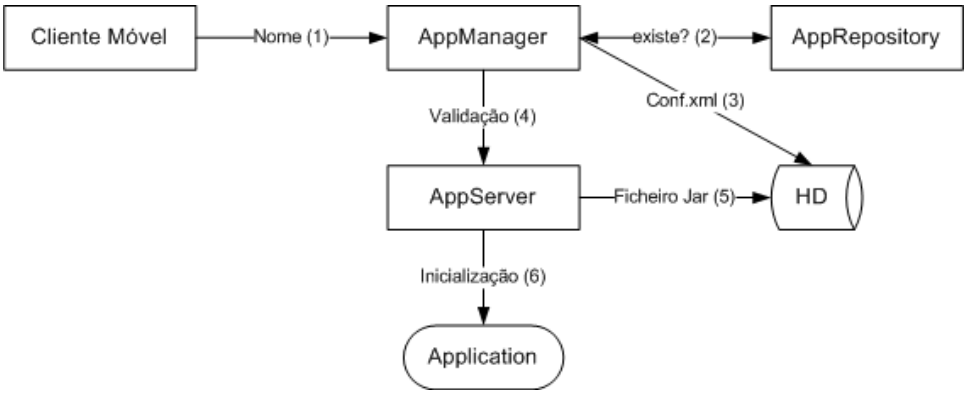

Figura 4.2: Processo de inicialização de uma aplicação de infra-estrutura.

Execução de aplicações definidas pelo utilizador

O cliente móvel mantém uma listagem das aplicações, das quais pode solicitar execução na infra-estrutura. Apesar de não ser obrigatório ter o seu código, é necessário que disponha dos respectivos ficheiros de configuração XML.

O processo de execução destas aplicações, apresentado na figura 4.3, inicia-se escolhendo uma aplicação do conjunto de aplicações locais. Para a aplicação seleccionada, transfere-se do cliente móvel para a infra-estrutura o respectivo ficheiro de configuração relativo à aplicação seleccionada (1). O ficheiro de configuração é validado (2) e é feito um pedido de inicialização da aplicação ao *AppServer*, que é responsável pela obtenção do arquivo com o código especificado (3) que pode residir no cliente móvel, ou em algum servidor na Internet. Finalmente, quando este ficheiro é obtido pode ser iniciada a aplicação através do processo descrito anteriormente.

4.2.3 Gestor de serviços

O gestor de serviços (figura 4.4) concentra as funcionalidades de gestão de serviços de um Infrastructure Node. Este módulo é composto por duas componentes. Uma primeira componente, que implementa a interface *ServiceManager* (listagem 4.4), é responsável pelo controlo dos serviços que correm na plataforma, incluindo o seu registo

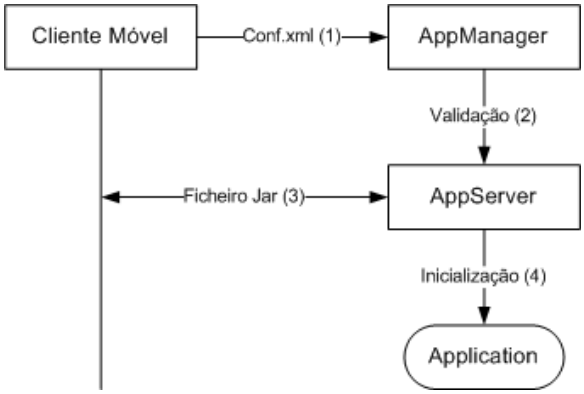

Figura 4.3: Processo de inicialização de uma aplicação definida pelo utilizador.

junto do directório de serviços. Uma segunda componente, o *ServiceLoader*, que por missão efectuar a inicialização dos diferentes serviços.

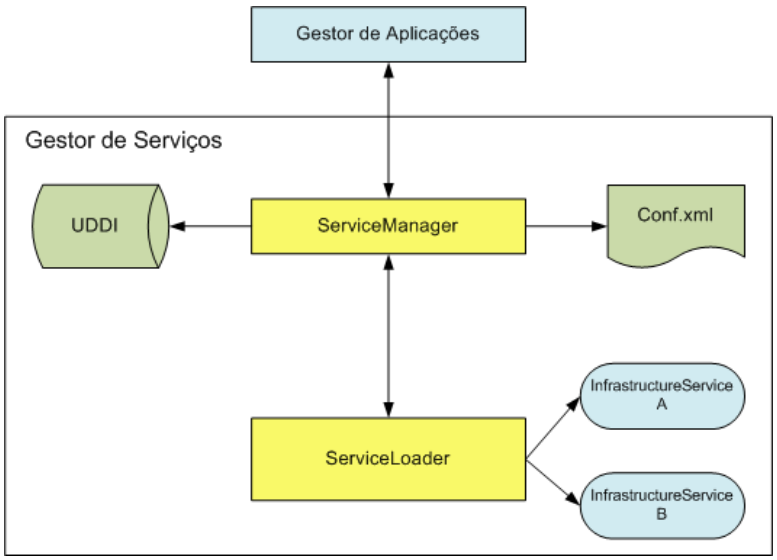

Figura 4.4: Estrutura interna do gestor de serviços.

Listagem 4.4: Interface ServiceManager

```
1 public interface ServiceManager extends BasicServiceManager{
2     public void addService(ServiceRecord sr);
3     public void stopService(String name);
4     public void addServices(List<ServiceLocator> list);
5 }
```

As operações fornecidas pela interface *ServiceManager* permitem efectuar a gestão do ciclo de vida de um serviço. O método *addService*, permite adicionar um novo serviço, efectuando a sua inicialização e registo junto do directório de serviços. O

método *addServices*, efectua o mesmo processo, para um conjunto de serviços. A operação *stopService*, permite parar um serviço, invocando o seu método de paragem e removendo a referência para o serviço em causa do directório de serviços.

Adicionalmente, esta interface estende a interface *BasicServiceManager*, que proporciona métodos para efectuar a instalação de serviços (*installService*) e para obter o respectivo endereço (*getServiceBindingUrl*).

A inicialização de um serviço tem bastantes semelhanças com o processo de inicialização de uma aplicação. Para tal, é usado o *ServiceLoader*, que começa por criar um contexto protegido onde o serviço corre. Seguidamente, é criada uma instância da classe que é definida como ponto de entrada do serviço, e que deve implementar a interface *InfrastrcutureService*. O novo serviço é devidamente configurado e finalmente, é invocado o método *startService*, que sinaliza que o serviço está devidamente configurado e pronto a correr.

Quando o processo de inicialização do serviço termina, este é registado pelo *ServiceManager* junto do directório UDDI. Na nossa implementação, para este papel utilizámos o jUDDI[3], que é uma implementação de um directório UDDI, totalmente em Java.

Como veremos na secção 4.2.4, na configuração de cada nó é especificado o seu nome único e a organização a que este pertence. Um serviço é registado como pertencente a uma dada organização, sendo guardado o URL do seu WSDL bem como no campo da descrição, o nome do nó onde este executa.

Esta informação é importante pois pode ser usada quando é feita a associação entre o serviço e o código cliente do mesmo. Por exemplo no contexto de uma aplicação que faça uso de um serviço câmaras, pode fazer sentido utilizar esta funcionalidade para escolher qual a câmara mais adequada.

Binding do cliente com o serviço

Uma aplicação ou um serviço pode fazer uso de outros serviços já existentes no sistema. Para tal, tanto a interface *InfrastructureApp*, como a interface *InfrastructureService*, apresentam um método que permite passar para a aplicação, ou para o serviço a referência para o gestor de serviços do Infrastructure Node.

Este objecto pode depois ser utilizado para obter o URL do WSDL de um serviço, de forma a que este possa ser utilizado no contexto de uma aplicação ou serviço. Tipicamente, no código da aplicação (ou serviço), já existe o código cliente desse serviço, pelo que este apenas é associado ao serviço obtido de forma a ser inicializado.

[3]http://ws.apache.org/juddi/

Instalação de novos serviços

As aplicações ou serviços podem fazer uso de outros serviços que não estejam disponíveis na infra-estrutura. Esta situação é detectada quando a aplicação tenta obter o URL para o WSDL de um serviço que não existe. Nestas circunstâncias, tipicamente é utilizada a informação que consta no objecto *ServiceLocator* para efectuar a sua instalação.

O *ServiceLocator* é um objecto criado a partir do ficheiro de configuração XML associado ao serviço ou aplicação, e que contém uma lista de URLs a partir dos quais é possível obter quer o ficheiro de configuração do serviço, quer o arquivo jar onde se encontra o respectivo código.

Com base nestes elementos, a aplicação pode solicitar ao gestor de serviços a instalação do novo serviço. Esta operação é efectuada através do método *installService* que é disponibilizado pela interface *BasicServiceManager*.

4.2.4 Módulo de configuração

O módulo de configuração permite definir a configuração do Infrastructure Node, incluindo as aplicações e serviços que estão disponíveis inicialmente.

No protótipo actual, a configuração do sistema é definida num ficheiro XML, em que se define de forma declarativa os diferentes parâmetros do sistema. No anexo A.2 apresenta-se o DTD relativo a este ficheiro de configuração.

As opções de configuração oferecidas podem-se agrupar em três grandes áreas: uma relativa à informação de administração do infrastructure node, outra relativa aos serviços que são fornecidos inicialmente, e finalmente uma secção relativa às aplicações que podem ser disponibilizadas.

Em termos da informação de administração, é definido o nome do nó, a organização a que pertence, alguma informação sobre endereços e portas e os dados que permitem aceder ao directório de serviços.

Relativamente às aplicações e serviços, em ambos os casos é apenas dada a localização do arquivo jar e do ficheiro de configuração, sendo esta a informação necessária à sua inicialização. No caso das aplicações podemos ainda ter algumas opções que permitem definir quais são aquelas que arrancam automaticamente e quais podem ser invocadas pelos utilizadores.

4.2.5 Implementação do modelo de segurança

Como foi discutido na secção 3.3, do ponto de vista da segurança devem ser resolvidos dois problemas distintos. O primeiro prende-se com a protecção da infra-estrutura

face à execução de código arbitrário. O segundo prende-se com o controlo de acessos de uma aplicação relativamente aos serviços que pode utilizar.

No protótipo actual, apenas a primeira questão é contemplada. A protecção da infra-estrutura relativamente à execução de código arbitrário é feita através das funcionalidades que a linguagem Java oferece, e traduz-se na conjugação de dois mecanismos distintos. O primeiro, visa garantir que as várias aplicações e serviços correm num contexto protegido, e o segundo visa associar diferentes permissões de segurança tendo em consideração se a aplicação ou serviço pertence à própria infra-estrutura ou a um utilizador.

Relativamente ao isolamento, cada aplicação está associada a um *classloader* "privado", que apenas conhece o código da aplicação para além do código das bibliotecas nativas do Java e mais algumas que sejam suportadas pela infra-estrutura (e.g. bluecove). Na figura 4.5 podemos ver o processo de instanciação de uma nova aplicação/serviço.

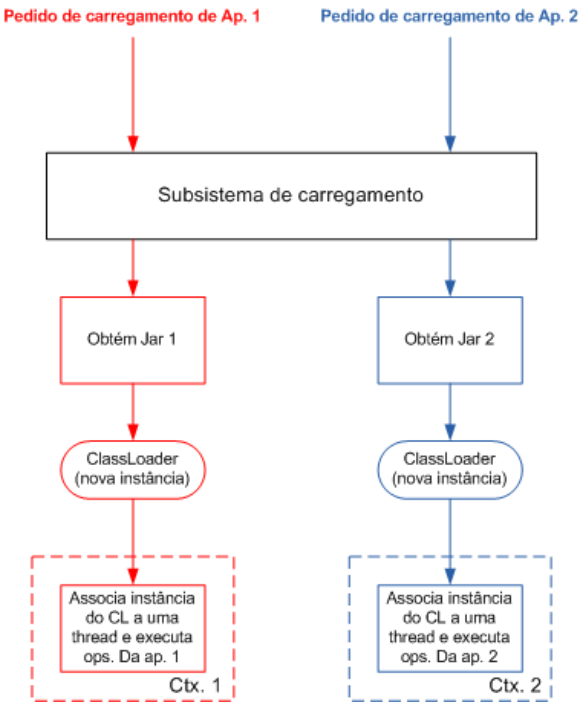

Figura 4.5: Criação de contextos protegidos para diferentes aplicações.

Associado a este mecanismo, existe um mecanismo que permite dotar as diferentes aplicações/serviços de permissões diferentes. Isto é feito associando às classes de cada contexto um conjunto de permissões diferentes. Na nossa implementação é suportada a diferenciação entre aplicações e serviços de infra-estrutura que correm com as permissões por omissão do Java, sendo considerados "de confiança"e aplicações e serviços de um utilizador (não autenticado), que correm com menos permissões. Es-

tas permissões permitem por exemplo, limitar o acesso à Internet, ou evitar que uma aplicação aceda a ficheiros fora da sua directoria temporária.

4.3 Cliente móvel

4.3.1 Módulo de comunicação e emparelhamento

Este módulo é análogo ao módulo de comunicação presente no infrastructure node. O seu objectivo é abstrair os restantes componentes da aplicação móvel dos detalhes relativos a este processo, permitindo que a comunicação seja vista de forma transparente independentemente da tecnologia utilizada.

Na nossa implementação são suportadas comunicações através de bluetooth e IP. Do ponto de vista operacional, este módulo permite efectuar o emparelhamento com o Infrastructure Node, e dá suporte às trocas de mensagens que permitem a execução de aplicações de infra-estrutura ou instalação de aplicações definidas pelo utilizador.

O processo de emparelhamento é responsável por obter os endereços de contacto dos diversos serviços disponibilizados pelo Infrastructure Node. Na nossa implementação este processo é realizado exclusivamente através de bluetooth de forma a garantir que o nó com o qual o dispositivo se emparelha está fisicamente próximo do utilizador. Os endereços obtidos são usados no contexto das restantes trocas de mensagens com a infra-estrutura.

As trocas de mensagens que suportam as funcionalidades de instalação de aplicações definidas pelo utilizador, ou de execução de aplicações de infra-estrutura, já foram apresentadas anteriormente, pelo que será focado o emparelhamento do cliente móvel com o Infrastructure Node.

O processo de emparelhamento em si é relativamente simples. Inicialmente a aplicação (móvel) começa por mostrar uma lista dos dispositivos bluetooth com o serviço de emparelhamento de infra-estrutura disponível. Após o utilizador efectuar a escolha de um elemento dessa lista é feito o processo de emparelhamento propriamente dito. Este processo também pode ser automático, sendo o emparelhamento efectuado com o primeiro Infrastructure Node disponível.

O processo de emparelhamento consiste numa troca de mensagens, em que o cliente móvel começa por enviar para a infra-estrutura uma mensagem do tipo *InfoRequestMessage*, que é respondida por uma mensagem do tipo *InfoReplyMessage*. Esta mensagem contém os endereços relativos aos serviços de execução e instalação de aplicações que o Infrastructure Node oferece, cada um destes serviços pode ser acedido por bluetooth e IP.

4.3.2 Gestor de Aplicações

O gestor de aplicações centraliza o processo de gestão e lançamento de aplicações a partir do cliente móvel. Este é directamente responsável pelo lançamento de aplicações da infra-estrutura e pela instalação de novas aplicações no Infrastructure Node.

Através do gestor de comunicações, este módulo obtém a listagem das aplicações de infra-estrutura que estão disponíveis para executar. Esta listagem, assume a forma de um documento XML onde se encontra a informação relativa às diversas aplicações que podem ser iniciadas. Essa informação consiste no identificador único da aplicação, isto é, o seu nome, bem como quais são as interfaces de controlo que esta exporta (web ou Java 2 ME) e uma descrição.

A aplicação seleccionada é guardada localmente num directório de aplicações, designado por *AppRegistry*. Para cada entrada neste repositório é mantido o identificador da aplicação e o tipo de meio de controlo associado. Caso este meio de controlo seja uma aplicação Java 2 ME, então é também mantida a informação sobre o URL onde é possível obter o respectivo arquivo jar e o respectivo URL de inicialização. Isto é útil na situação em que aplicação em causa requer outra aplicação Java 2 ME, que terá de ser transferida e instalada. Através deste registo é possível verificar se essa aplicação já foi instalada e caso isto se verifique iniciá-la imediatamente.

Em relação à funcionalidade de instalação de aplicações na infra-estrutura, esta é suportada por um directório, mantido internamente pelo cliente móvel, onde constam as aplicações que podem ser transferidas para a infra-estrutura. Este consiste numa directoria que serve como repositório das aplicações que podem ser instaladas. Nesta directoria constam obrigatoriamente os ficheiros XML de configuração, podendo ainda constar os respectivos arquivos jar.

Independentemente do tipo de aplicação que se trata, ao utilizador é apresentada apenas uma lista com todas as aplicações que este pode executar. Os detalhes de inicialização dos diferentes tipos de aplicações, são tratados internamente, sendo totalmente transparente para o utilizador.

Um aspecto bastante importante neste módulo e comum a ambas as funcionalidades descritas anteriormente, tem a ver com o tratamento dos meios de controlo associados às aplicações que correm na infra-estrutura.

No caso das interfaces Web, quando um pedido desta natureza é recebido, é utilizado o browser do telemóvel para mostrar a página referenciada pelo URL através da utilização do método *platformRequest* que está disponível às aplicações Java 2 ME.

Caso se trate de uma aplicação móvel que é descarregada, após o respectivo arquivo jar ter sido recebido pelo gestor de comunicações, procede-se à sua instalação. Na prática é usado o método *platformRequest*, que leva como argumento o caminho para o

ficheiro jar que se encontra agora armazenado no dispositivo móvel.

Para que a aplicação possa ser iniciada, é efectuado o seu registo junto do Push Registry[4] do equipamento móvel. Tipicamente, isto é feito durante o próprio processo de instalação (através do atributo MIDlet-Push que pode ser definido no ficheiro manifest do jar). Caso não seja, é necessário correr a nova aplicação de forma a efectuar o seu registo através das opções que esta disponibilize. O Push Registry faz parte do sistema de gestão de aplicações do dispositivo móvel, e permite que uma aplicação seja iniciada automaticamente, por exemplo através de um contacto num socket.

Uma vez que nos equipamentos que utilizámos não é possível notificar uma aplicação de quando a instalação de outra é concluída, quando o processo de instalação de uma aplicação é iniciado o cliente móvel é bloqueado, ficando à espera da confirmação do utilizador que a nova aplicação está instalada. Quando esta parte do processo é concluída, então o cliente móvel usa a forma de activação da aplicação especificada no ficheiro de configuração para a iniciar. Geralmente esta activação consiste no contacto com uma socket numa porta específica.

[4]http://developers.sun.com/mobility/midp/articles/pushreg/

5

Aplicações e serviços de teste

A solução apresentada ao longo deste documento visa a criação de uma infra-estrutura genérica, que permita a execução de diferentes aplicações que façam uso de serviços disponibilizados num ambiente de computação ubíqua. Como tal, uma parte importante da validação deste sistema consiste no desenvolvimento de diferentes aplicações e serviços que façam uso das diversas funcionalidades oferecidas. Neste capítulo apresentam-se os diversos serviços que foram implementados, seguindo-se as aplicações de demonstração criadas e que fazem uso dos referidos serviços.

5.1 Serviços

5.1.1 Serviço de detecção bluetooth

O objectivo do serviço de detecção bluetooth é concentrar as funcionalidades de detecção de dispositivos bluetooth para que possam ser utilizadas por diversas aplicações, sem que estas se preocupem com os detalhes de implementação deste processo.

Este serviço é exportado como um Web Service SOAP, e foi implementado usando as funcionalidades que o Java 1.6 fornece para o efeito. A detecção bluetooth faz uso da biblioteca Bluecove [1]. Este tem apenas um método acessível aos clientes, o método *probe*:

```
List<DeviceEntry> probe(long[] uuids, int[] devClass);
```

[1] http://www.bluecove.org/

Este método recebe dois arrays que correspondem às classes de dispositivos passíveis de ser pesquisadas (devClass) e os UUIDs dos serviços a pesquisar (uuids). Como resultado, retorna uma lista de objectos do tipo DeviceEntry que guardam o endereço bluetooth obtido, a força de sinal e o endereço do serviço pesquisado.

Relativamente à força de sinal, esta apenas está disponível em nós Linux, nos quais é possível correr um script Python que é responsável por obter esta informação[2]. A utilização de código externo ao Java, como neste caso, foge ao controlo dos mecanismos de segurança oferecidos pelo sistema. Como tal, serviços que necessitem de scripts ou código noutras linguagens devem correr com o nível de segurança mais elevado, isto é, como pertencentes à própria infra-estrutura.

5.1.2 Serviço de câmaras

O serviço de câmaras permite que as aplicações obtenham dados de uma câmara associada ao infrastructure node onde o serviço corre. Os dados obtidos podem assumir duas formas distintas: "snapshots"ou streams de vídeo. Este serviço é disponibilizado por meio de Web Services SOAP, sendo usado o protocolo RTP para efectuar a transmissão da stream de vídeo. Internamente a este serviço, o contacto com a câmara é feita através das funcionalidades oferecidas pelo JMF [3]. O serviço exporta os seguintes métodos:

```
byte[] getSnapshot();
boolean startStream(String ip, int port);
void stopStream();
```

O método *getSnapshot* permite efectuar a captura de uma imagem da câmara em formato jpeg.

Os métodos *startStream* e *stopStream* são utilizados no âmbito do envio de uma stream de vídeo para o cliente. Como já foi referido, este processo é efectuado através da utilização do protocolo RTP, sendo a componente de Web Services responsável por solicitar o início e o fim da emissão. Mais concretamente, o método *startStream* é responsável por notificar o serviço de câmaras que há um cliente que requer uma stream de vídeo, indicando o seu endereço IP e porta a usar. Internamente, usando as capacidades oferecidas pelo JMF é iniciada a stream destinada ao cliente.

Na situação em que o cliente decide terminar a stream de vídeo é utilizado o método *stopStream* que faz com que o servidor RTP termine e liberte os recursos alocados.

[2]Baseado no exemplo de: http://wiki.bluez.org/wiki/HOWTO/DiscoveringDevices
[3]http://java.sun.com/javase/technologies/desktop/media/jmf/

5.1.3 Serviço de detecção de faces

O serviço de detecção de faces, permite detectar as caras existentes numa imagem, podendo ainda indicar quantas correspondem a indivíduos do sexo masculino ou feminino.

Este serviço consiste num wrapper Java(JNI) para a biblioteca C++ de reconhecimento de faces implementada por Grangeiro et. al [GJC09]. Neste caso concreto o serviço apenas está disponível em Infrastructure Nodes que estejam a correr sistemas operativos Windows (XP ou Vista), sendo disponibilizados através de Web Services SOAP os seguintes métodos:

```
long computePicture(byte[] pic);
int getNumPeople(long picId);
int getNumMales(long picId);
int getNumFemales(long picId);
```

O método *computePicture* pode ser invocado pelo cliente de forma a enviar a imagem a processar a este serviço, devolvendo ao cliente um identificador único. Este identificador pode ser usado nos restantes métodos de forma a indicar qual a imagem a ser processada. Estes métodos para permitem obter informação sobre o número total de pessoas detectadas (*getNumPeople*) ou relativamente ao número de homens ou mulheres detectados (*getNumMales* e *getNumFemales* respectivamente).

5.1.4 Serviço de adaptação de conteúdos (*transcoding*)

O serviço de adaptação de conteúdos permite adaptar conteúdos multimédia, transformando a sua qualidade e a sua representação. No protótipo actual é possível efectuar a adaptação de imagens e efectuar a conversão e adaptação de vídeos. Esta última funcionalidade tem como base o conversor ffmpeg [4]. Este serviço exporta os seguintes métodos:

```
MediaWrapper transcodeMedia(String url, String targetFormat, Properties
    transformProps);
```

O método *transcodeMedia* permite fazer a adaptação de diferentes conteúdos multimédia. Este recebe o URL dos conteúdos a adaptar, bem como o formato alvo e o conjunto de propriedades que podem ser relevantes no contexto da adaptação. Estas propriedades variam consoante o tipo multimédia em causa. Por exemplo no caso de se tratar de um vídeo, estes podem ser as suas dimensões, o seu frame rate, o seu bit rate, entre outras. Quando o processo de transformação é concluído, os dados são

[4] http://ffmpeg.org/

enviados ao cliente que invocou este serviço sob forma de um objecto do tipo MediaWrapper.

Os objectos deste tipo encapsulam o resultado da transformação dos diferentes conteúdos multimédia, guardando não só os conteúdos propriamente ditos (sob a forma de um array de bytes) mas também alguma informação adicional (formato, nome do ficheiro). Estes objectos são independentes do tipo de multimédia podendo ser usados para transferir vídeos ou imagens para o utilizador.

5.1.5 Serviço de reprodução multimédia

A gestão dos acessos concorrentes a um ecrã público é uma questão importante num sistema como o PIPE. O serviço de reprodução de conteúdos multimédia permite escalonar os pedidos desta natureza dirigidos a um Infrastructure Node.

À semelhança dos restantes serviços já apresentados, este apresenta uma interface, disponibilizada através de Web Services SOAP, que tem os seguintes métodos:

```
long playMedia(String mediaLocation, String mime–type, String btAddress);
void stopMedia(long requestId);
```

Internamente, este serviço apresenta um módulo que é responsável pelo escalonamento e controlo dos conteúdos a reproduzir. Este módulo faz uso do serviço de pesquisa bluetooth descrito anteriormente de forma a ser possível associar cada pedido a um dispositivo bluetooth.

Como tal, no método *playMedia* é possível especificar, além do URL para o conteúdo multimédia e do respectivo tipo, o endereço bluetooth do dispositivo associado a este pedido. O objectivo é permitir um modelo de reprodução presencial, isto é, os conteúdos são reproduzidos enquanto o dispositivo associado ao pedido for detectado. Caso o parâmetro que indica o endereço bluetooth seja *null*, então o conteúdo especificado é reproduzido até ao fim. O método *playMedia* devolve um identificador do pedido, que pode ser usado pela operação *stopMedia* para parar a reprodução, ou remover o pedido.

A componente que trata da reprodução dos conteúdos foi implementada usando as capacidades oferecidas pela JMF. Para permitir a reprodução de conteúdos multimédia sem uma dimensão temporal, como por exemplo imagens, na implementação actual estipula-se um tempo máximo de apresentação.

No protótipo actual, este serviço usa uma disciplina FIFO para escalonar os diversos pedidos, contudo não são de excluir outras disciplinas de ordenação. Em particular, seria interessante explorar ordenações que privilegiem utilizadores que paguem pelo serviço em detrimento dos restantes.

Também seria interessante explorar outras formas de visualização de acessos concorrentes ao ecrã, por exemplo através da atribuição de *time-slots* a cada pedido, ou através da divisão do ecrã em várias regiões, cada qual a mostrar um pedido distinto.

5.2 Aplicações

Nesta subsecção apresentaremos duas aplicações demonstrativas da utilização do sistema desenvolvido.

5.2.1 Aplicação de publicidade

A primeira aplicação tem como objectivo apresentar publicidade numa zona comercial. Para tal, assume-se que existe um conjunto de Infrastructure nodes distribuídos por essa zona, bem como alguns ecrãs que podem ser usados para apresentar informação e que são controlados pelo sistema. Adicionalmente, existem câmaras de vigilância que cobrem a zona comercial e que podem ser usadas pelo sistema.

A publicidade pode ser mostrada de duas formas distintas, num ecrã público associado ao Infrastructure Node que corre a aplicação, ou directamente no telemóvel do utilizador. A ideia base é que estas duas técnicas se complementem. Numa situação em que são detectados poucos utilizadores (e.g. 2 ou 3), pode não ser desejável mostrar a publicidade no ecrã público, já que a sua privacidade pode ser comprometida, pois é fácil perceber a quem é que o anúncio mostrado se dirige. Nesta situação, consideramos que pode ser desejável enviar directamente os conteúdos para o dispositivo móvel do utilizador. Actualmente os conteúdos mostrados assumem a forma de imagens estáticas, mas a aplicação está preparada para utilizar outros formatos multimédia.

A aplicação de publicidade utiliza três serviços distintos: o serviço de publicidade, o serviço de adaptação e o serviço de reprodução multimédia.

O serviço de publicidade permite que a aplicação obtenha informação dos dispositivos (e utilizadores) próximos do nó da infra-estrutura, dando conhecimento das respectivas preferências. Isto possibilita que seja efectuado o processo de escolha dos anúncios a mostrar num ecrã público ou a enviar directamente para os diversos dispositivos.

Caso os anúncios estejam para ser enviados para o equipamento móvel dos utilizadores, a aplicação recorre ao serviço de adaptação de conteúdos, de forma a adaptar a sua qualidade antes de iniciar a transferência para um dispositivo móvel.

Esta aplicação é exemplificativa de uma aplicação que corre na infra-estrutura, sem necessidade duma interacção explícita por parte do utilizador. Adicionalmente,

demonstra a criação duma aplicação que utiliza um conjunto variado de serviços, alguns dos quais executam necessariamente num determinado Infrastructure Node (e.g. serviço de publicidade e serviço de reprodução de multimédia), enquanto outros podem executar em qualquer outro nó do sistema (e.g. serviço de adaptação).

5.2.2 Serviço de publicidade

O serviço de publicidade destina-se a fornecer à aplicação de publicidade um serviço que permita obter informações relativas aos utilizadores e suas preferências. Este serviço vem também demonstrar a capacidade de efectuar uma composição de serviços de forma a permitir que sejam fornecidos serviços de mais alto nível às aplicações.

Neste caso é usado o serviço de detecção bluetooth, o serviço de câmaras e o serviço de detecção de faces. Com base nestes três serviços, é possível não só ter uma noção do número de utilizadores que se encontram nas proximidades do infrastructure node, mas também ter ideia de quantos destes são homens ou mulheres.

Este serviço conta ainda com uma base de dados, que na nossa implementação está centralizada num servidor externo ao serviço de publicidade.

A base de dados guarda a informação relativa aos utilizadores e locais, e pode ser acedida e actualizada através das operações disponibilizadas pelo serviço de publicidade. O objectivo é facilitar a partilha de informação entre diversos nós que executem este serviço.

A estrutura da base de dados é apresentada na figura 5.1. Em traços gerais, utilizam-se os dispositivos bluetooth como *proxies* para os utilizadores humanos, sendo cada pessoa identificada pelo endereço bluetooth do seu equipamento móvel (tabela *users*). Adicionalmente, são guardados os locais onde os utilizadores são detectados (tabela *history*). Cada local está associado a um conjunto de tags, que definem quais são os interesses associados a esse espaço físico (e.g. um local próximo de uma livraria pode ter tags como "livros"ou "cultura")(tabela *location*). Com base no histórico das deslocações dos utilizadores no espaço e as tags associadas aos diferentes locais é possível inferir as suas preferências. Estas preferências assumem a forma de um conjunto de tags, estando cada tag associada a um valor numérico, o seu peso (tabela *userPrefs*). Quando maior for este valor numérico, maior será o interesse do utilizador em determinado tópico.

Este serviço à semelhança dos restantes assume a forma de um Web Service, o qual exporta os seguintes métodos:

```
List<UserInfo> getNearbyUsers();
VisualDetectionInfo getCameraInfo();
```

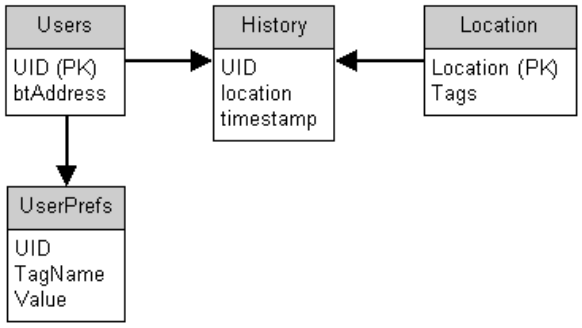

Figura 5.1: Esquema da base de dados subjacente.

O primeiro método (*getNearbyUsers*) permite obter a lista dos utilizadores detectados nas proximidades do Infrastructure Node, usando para tal o serviço de detecção bluetooth. Adicionalmente, este método actualiza as informações relativas às preferências de cada utilizador. A informação devolvida consiste numa lista de objectos do tipo *UserInfo*. Cada objecto deste tipo, encapsula a informação relativa a um utilizador, ou seja resultado da leitura bluetooth (endereço do dispositivo, força de sinal e endereço do serviço de transferência de ficheiros através de OBEX [obe09]) e a respectiva lista de interesses. Esta lista é composta por diversas entradas, cada qual formada por uma tag e por um valor numérico que é o seu peso. Este valor aumenta à medida que utilizador é detectado por mais tempo no mesmo local, ou em locais com a mesma tag.

O método *getCameraInfo* permite obter dados combinando o serviço de câmaras com o serviço de detecção de faces. A informação obtida, número de homens e número de mulheres detectados é devolvida no resultado da operação.

Funcionamento da aplicação

Como já foi referido, assume-se que a aplicação de publicidade se encontra a correr num conjunto de nós espalhados pelo espaço. Cada nó está associado a um conjunto de tags que indicam os interesses associados a esse local.

Na nossa implementação, a detecção de utilizadores é feita com base no serviço de publicidade, o que permite à aplicação obter informação relativa aos utilizadores detectados.

Essa informação contém a lista de interesses de cada utilizador e é usada como base para o processo de escolha dos anúncios. Adicionalmente, a aplicação pode fazer uso das funcionalidades que o serviço de publicidade oferece no âmbito da utilização de câmaras, para obter alguma informação adicional sobre o número e sexo dos indivíduos detectados.

Os anúncios são conteúdos multimédia que estão associados também a diversas tags, e o processo de escolha dos anúncios é feito com base na comparação entre os interesses dos utilizadores detectados, com os anúncios existentes, procurando escolher-se aqueles que se adequam melhor ao público presente. Seguidamente descreve-se este processo.

Processo de escolha dos anúncios:

Na prática existem duas formas distintas de efectuar a escolha dos anúncios. A primeira relativa aos casos em que os conteúdos são enviados directamente para o dispositivo móvel. A segunda para quando os anúncios são mostrados num ecrã público.

Relativamente ao primeiro caso, o processo é bastante simples, sendo escolhido o anúncio (que ainda não tenha sido enviado) que se adequa melhor aos interesses do utilizador. Isto é conseguido cruzando a lista de interesses do utilizador (tags e respectivos pesos) com as tags dos vários anúncios disponíveis. O anúncio escolhido é aquele cujas tags somam um maior peso tendo em conta os interesses do utilizador em causa. Os anúncios são enviados através do protocolo OBEX sobre bluetooth.

A tabela 5.1 mostra as preferências de um utilizador, face aos anúncios existentes, exibindo igualmente a ordem pela qual esses anúncios seriam enviados ao utilizador.

Utilizador		**Anúncios**	
Tag	Peso	Nome	Tags
technology	9776	A1	books
books	6410	A2	books, technology
movies	3366	A3	food
food	3366	A4	Gardening
Anúncios seleccionados			
1º A2		9776 + 6410	
2º A1		9776	
3º A3		3366	
4º A3		0	

Tabela 5.1: Interesses de um utilizador e anúncios disponíveis

Nesta situação, o envio de publicidade não solicitada para os dispositivos dos utilizadores deve ser tida em consideração. Como tal, sempre que o envio de um anúncio falha, assume-se que tenha sido porque o utilizador o rejeitou. Em qualquer caso, fica um período de tempo sem contactar esse dispositivo antes de efectuar novamente o envio de publicidade.

No segundo caso, em que os anúncios são apresentados num ecrã público, é feito um agregado dos pesos das várias tags correspondentes aos interesses dos utilizadores detectados. Seguidamente, os valores agregados são cruzados com as tags dos anúncios, sendo escolhido o anúncio cujo somatório dos pesos das suas tags dê um maior valor.

No nosso protótipo, nas iterações seguintes esse anúncio não é mostrado, pelo menos até todos os outros anúncios terem sido mostrados. Uma alternativa interessante a este esquema foi explorada no contexto do sistema Bluscreen [KPD07], no qual a cada anúncio estava associado um orçamento, que ia sendo gasto à medida que o anúncio era mostrado. Quando esse orçamento era esgotado, o anúncio em causa era efectivamente descartado.

5.2.3 Aplicação de informação estendida sobre produtos

A aplicação de informação estendida sobre produtos permite ao utilizador obter um conjunto de informações relativas a um produto com base no seu código de barras. Esta aplicação recorre a um conjunto de fontes de dados na Web, em particular os serviços da Amazon [5] e do YouTube [6], oferecendo um agregado (*mashup*) desses dados ao utilizador. Neste agregado consta não só a informação relativa ao produto em causa, mas também informação sobre produtos e videos relacionados.

Do ponto de vista do utilizador, após este efectuar uma pesquisa, no cliente móvel tem acesso à informação relativa ao produto cujo código de barras foi capturado - figura 5.2-1.

A partir deste ponto, pode ter-se acesso à seguinte informação. Primeiro, uma lista dos vários produtos relacionados - figura 5.2-2. Escolhendo um desses itens é mostrada a informação detalhada desse produto (num ecrã semelhante ao primeiro).

Segundo, pode ter-se acesso a informação detalhada do vídeo que foi considerado como mais relevante na pesquisa efectuada junto dos serviços do YouTube - figura 5.2-3. O utilizador pode fazer a sua visualização.

Terceiro, é possível aceder-se a uma lista de vídeos relacionados. A selecção de um elemento dessa lista, leva o utilizador para um ecrã semelhante ao terceiro, onde pode efectuar a sua visualização - figura 5.2-4.

Esta aplicação visa demonstrar a viabilidade de aplicações que corram as partes computacionalmente mais exigentes na infra-estrutura.

A aplicação móvel foi construída em Java 2 ME, e pode subdividir-se em três grandes módulos: captura, pesquisa e agregação.

O módulo de captura, faz uso da biblioteca ZXing[7] para efectuar o reconhecimento dos códigos de barras. O módulo de pesquisa, recebe a informação do módulo anterior e com base nas preferências do utilizador contacta o módulo de agregação, que é responsável pela obtenção dos dados relativos ao produto e vídeos relacionados.

[5] http://docs.amazonwebservices.com/AWSEcommerceService/2005-03-23/
[6] http://code.google.com/intl/pt-PT/apis/youtube/overview.html
[7] http://code.google.com/p/zxing/

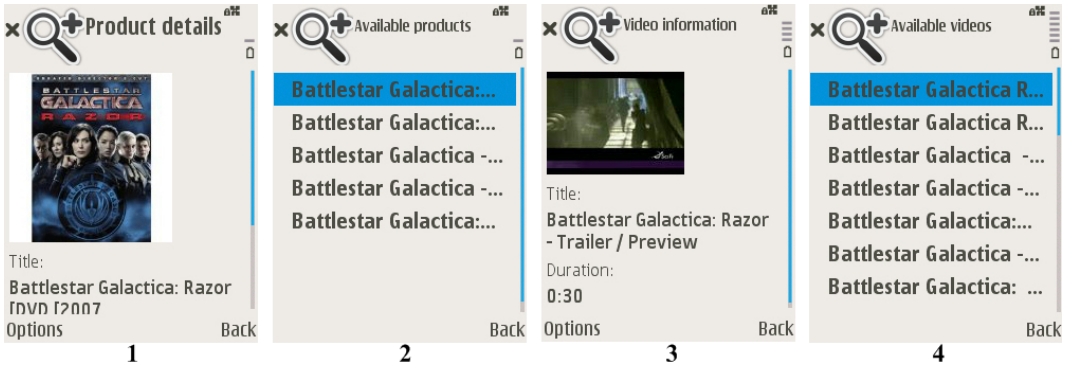

Figura 5.2: Screenshots da aplicação em funcionamento.

Esta aplicação exibe dois perfis de funcionamento, consoante o módulo de agregação execute localmente no dispositivo móvel (modo *stand-alone*), ou na infra-estrutura (modo de infra-estrutura). Independentemente de onde este módulo se encontre a executar, este processa os pedidos relativos aos produtos detectados. Para tal, contacta os serviços da Amazon para obter informação relativa ao produto em causa, incluindo os produtos relacionados, e os serviços do YouTube para obter vídeos relacionados.

Adicionalmente, é possível ao utilizador, efectuar o download dos diferentes vídeos mostrados, ou, caso a aplicação se encontre a operar em modo de infra-estrutura, visualizá-los num ecrã público. Nesta última situação, continua a ser possível efectuar o download do vídeo, mas a infra-estrutura faz a adaptação dos conteúdos para um formato adequado ao dispositivo móvel em causa, usando o serviço de adaptação.

Este processo está esquematizado na figura 5.3

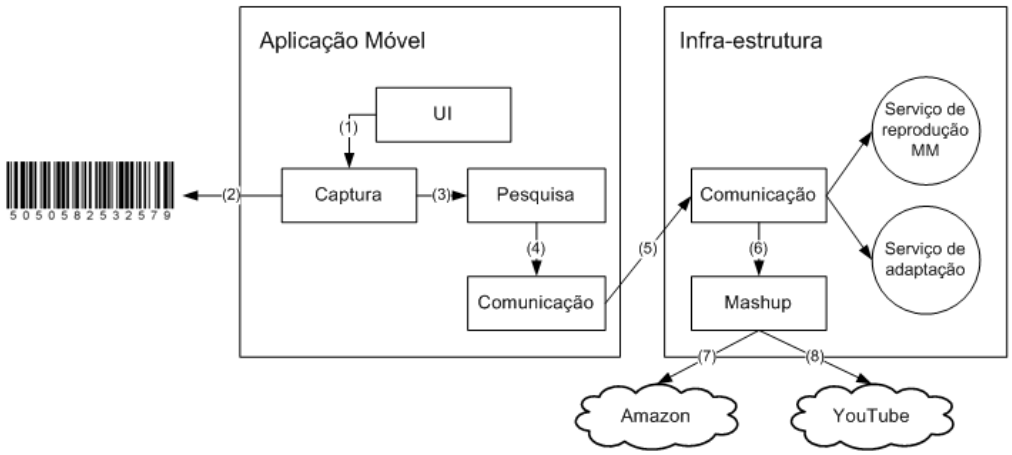

Figura 5.3: Interacção da componente móvel da aplicação com a componente de infra-estrutura.

Funcionamento da aplicação

Nesta aplicação, a componente de agregação pode executar no Infrastructure Node. Quando tal situação se verifica, esta componente pode ser instalada a partir do dispositivo móvel do utilizador. Adicionalmente, caso o serviço de adaptação não esteja disponível este pode ser também instalado (isto requer que o Infrastructure Node tenha o conversor ffmpeg disponível).

Em termos do funcionamento propriamente dito, após ser capturada a informação do código de barras do produto, é pedido ao utilizador que identifique que tipo de produto se trata (um filme, música ou software). Este passo é importante na medida em que vai influenciar os parâmetros de busca que são usados junto dos serviços do YouTube (e.g. se for um filme na pesquisa para além do seu nome é também usada a string "movie trailer").

Independentemente do contacto com as diversas fontes de dados ser feito na infra-estrutura ou no próprio dispositivo móvel, a informação obtida consiste no identificador do produto (o valor constante no seu código de barras), o respectivo nome, uma imagem (tipicamente a capa), o valor da classificação média bem como uma listagem de restantes produtos relacionados. Cada produto relacionado apenas contém o seu identificador e o respectivo título.

Obtidos estes dados, é possível usar o título do produto e os parâmetros adicionais de pesquisa para efectuar a busca junto dos serviços do YouTube. Esta pesquisa é ordenada por relevância, e o primeiro resultado é guardado como vídeo principal. Isto implica que é guardado o seu título, a sua duração, uma imagem, bem como o endereço para efectuar o seu download e uma lista de vídeos relacionados. Relativamente a estes vídeos é apenas guardado o seu título e o respectivo identificador para permitir posteriores consultas. O processamento dos documentos XML obtidos destes dois serviços é feito através de um parser SAX, o que permite que a implementação desta funcionalidade na infra-estrutura não varie significativamente daquela que é realizada no cliente móvel. Este aspecto foi importante para facilitar posteriores comparações entre estes dois modos de operação.

Quando o utilizador decide fazer download de um vídeo, a aplicação exibe comportamentos diferentes, consoante esteja a operar em modo infra-estrutura ou em modo stand-alone.

No primeiro caso, dirige um pedido à componente que corre na infra-estrutura com o URL para download do vídeo. Esta efectua o seu download (em formato *flv*) e efectua a sua transformação no formato adequado ao dispositivo móvel em causa através do serviço de adaptação. Após esta adaptação ter sido feita, o vídeo é transferido de volta para a componente móvel que usa um *player* interno à aplicação para efectuar a

sua reprodução. Adicionalmente, todos os vídeos descarregados desta maneira são arquivados no histórico de vídeos de forma a poderem ser visualizados posteriormente.

No segundo caso, o tratamento é mais simples, sendo simplesmente feito um platformRequest com o URL do vídeo, usando-se o browser do dispositivo móvel para efectuar a sua reprodução.

Adicionalmente, caso a aplicação se encontre a operar em modo de infra-estrutura, o utilizador pode solicitar a visualização do vídeo directamente no Infrastructure Node, onde a componente de agregação se encontra a executar. Numa primeira fase, o vídeo em causa é descarregado para a infra-estrutura, ficando a sua reprodução a cargo do serviço de reprodução de multimédia discutido anteriormente.

Uma melhoria interessante que podia ser feita a esta aplicação, consiste em descarregar o vídeo mais relevante antecipadamente, de forma a melhorar o desempenho.

6

Resultados

Neste capítulo apresenta-se a avaliação realizada ao sistema PIPE. Esta consistiu num conjunto de testes que pode ser dividido em duas partes. Na primeira parte são mostrados alguns resultados de carácter mais geral do sistema. Na segunda parte são apresentados os resultados obtidos com a aplicação de informação estendida sobre produtos.

Os resultados apresentados nas várias tabelas que constam neste capítulo, correspondem à média de um conjunto de cinco medições.

6.1 Ambiente

Os testes que realizados recorreram apenas a um cliente móvel e a um Infrastructure Node. Como cliente móvel foi utilizado um telemóvel Nokia N82, com o sistema operativo Symbian OS 9.2 e um processador dual ARM 11. Como Infrastructure Node, usou-se um PC equipado com um processador AMD Turion 64x2, 2GB de RAM e Windows Vista. O computador estava equipado com um adaptador bluetooth (versão 2.0 EDR), um adaptador Wi-Fi (802.11b/g) e com uma ligação à Internet de 20Mbps, através de Wi-Fi.

Em diversos testes, foi necessário aceder a arquivos jar remotos, para subsequente instalação. Para tal, foi usado um servidor dedicado, cujo endereço IP é 94.23.47.54 e apresenta uma latência média de 32ms relativamente à rede onde foram executados os testes.

6.2 Instalação dinâmica de aplicações

Nesta secção apresentam-se os testes efectuados ao sistema PIPE. Estes testes avaliam o processo de transferência de aplicações entre o dispositivo móvel e a infra-estrutura e vice-versa. Finalmente, apresenta-se ainda um perfil da utilização de memória no cliente móvel.

6.2.1 Instalação de aplicação na infra-estrutura

Atendendo ao processo de instalação de uma aplicação definida pelo utilizador na infra-estrutura, abordado na secção 4.2.2, o objectivo deste teste foi avaliar o tempo de transferência de uma aplicação do cliente móvel, para a infra-estrutura. Para tal, efectuaram-se três medições distintas. A primeira consistiu na medição do tempo de transferência do ficheiro de configuração da aplicação ,a segunda, do tempo de transferência do arquivo com o respectivo código e a terceira, do tempo gasto com o carregamento e inicialização da aplicação na infra-estrutura.

Para esta aplicação, com aproximadamente 91 KB, constata-se que o tempo médio decorrido desde que esta é recebida até começar a executar é de 696,4 ms.

Esta experiência foi efectuada utilizando diferentes tecnologias de comunicação e variando a localização do arquivo com o código da aplicação. Nos dois primeiros testes, este residia no dispositivo móvel ao passo que no último teste este encontrava-se num servidor remoto. Os dados obtidos encontram-se agrupados na tabela 6.1. Note-se que no teste relativo à transferência do arquivo num servidor remoto, o ficheiro de configuração continua a ser enviado a partir do dispositivo móvel, e por isso optou por não se colocar esse valor na tabela.

		Bluetooth	**Wi-Fi**	**Wi-Fi (arquivo remoto)**
	Tamanho (bytes)	**Tempo médio (ms)**	**Tempo médio (ms)**	**Tempo médio (ms)**
Fich. config.	621	2629,4	2064	-
Arquivo jar	93278	3885	3109,2	529,2

Tabela 6.1: Tempos de instalação de uma aplicação na infra-estrutura

Este teste vem confirmar que o processo de instalação de aplicações é rápido, para a dimensão típica de uma aplicação. O tempo de instalação de uma aplicação na infra-estrutura não excede alguns segundos, por exemplo no caso em que é utilizado bluetooth, gasta-se em média 7,21 segundos até que a aplicação comece a executar. Este facto é importante tendo em conta o cenário para o qual este sistema está vocacionado.

Como seria de esperar a transferência de ficheiros através de Wi-Fi é mais rápida que através de bluetooth, sendo preferível usar este meio de comunicação para ficheiros

maiores.

6.2.2 Instalação de aplicação no cliente móvel

O segundo teste avalia o tempo de instalação de uma nova aplicação no dispositivo móvel. Uma vez que no equipamento utilizado este processo requer um conjunto bastante grande de autorizações por parte do utilizador, este teste foca exclusivamente o tempo de transferência do arquivo com o código da aplicação móvel. À semelhança do teste anterior, este foi efectuado em diversos cenários distintos, contemplando não só as várias tecnologias de comunicação suportadas, mas também a localização do código. Os resultados encontram-se na tabela 6.2.

		Bluetooth	Wi-Fi	Wi-Fi (arquivo remoto)
	Tamanho (bytes)	**Tempo médio (ms)**	**Tempo médio (ms)**	**Tempo médio (ms)**
Arquivo jar	315392	2800,6	956	1135

Tabela 6.2: Tempos de instalação de uma aplicação no cliente móvel

Neste teste, também se verifica que a de transferência de uma aplicação para o cliente móvel é rápida. Nesta experiência, uma vez que o ficheiro transferido é maior, as diferenças entre as várias tecnologias de comunicação são mais evidentes.

Comparando com o teste anterior, verifica-se que apesar do ficheiro usado neste teste ser substancialmente maior, com 308 KB, o processo de transferência é mais rápido. Este resultado, que é consistente para todas as tecnologias de comunicação abordadas (à excepção da localização remota), pode ficar a dever-se a aspectos de mais baixo nível, relacionados com a própria implementação dos mecanismos de comunicação. Uma hipótese interessante para explicar este resultado, está relacionada com o facto da transmissão de dados gastar bastante energia. Por isso, quando é o dispositivo móvel a transmitir, este transmite a um ritmo mais lento, para limitar o gasto de energia. No caso inverso, os equipamentos de infra-estrutura não têm de lidar com esse tipo de questões, e por isso podem transmitir à velocidade máxima.

6.2.3 Ocupação de memória

Neste teste avalia-se o consumo de memória inerente à utilização do cliente móvel. Para efectuar as medições utilizámos o software Nokia Energy Profiler no dispositivo móvel.

O teste em si consistiu no lançamento da aplicação do sistema, que permite o acesso ao Infrastructure Node seguido pelo pedido de execução da aplicação de informação estendida sobre produtos. A figura 6.1, mostra a evolução do consumo de memória ao

longo desta experiência. A linha a tracejado indica a memória ocupada pelo sistema operativo do telemóvel (28,95 MB), sendo os consumos de memória das aplicações calculados a partir desse valor base.

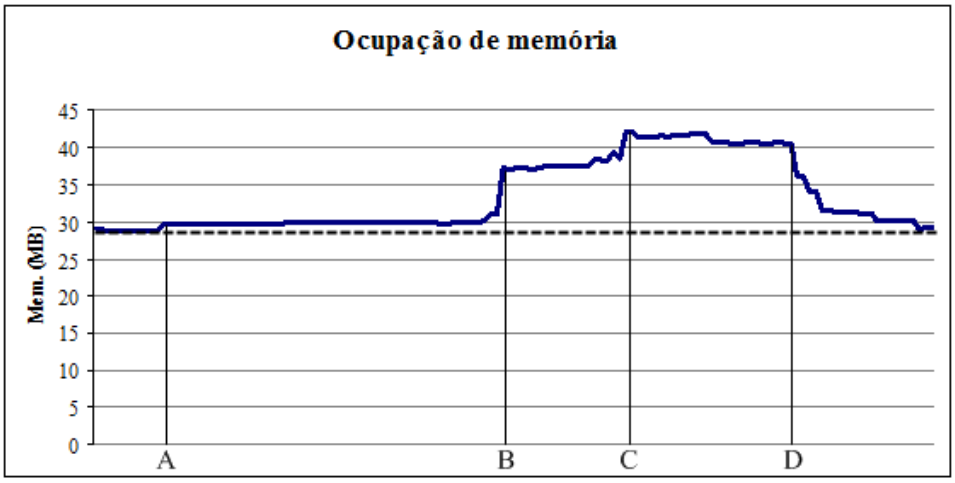

Figura 6.1: Evolução do consumo de memória ao longo do tempo.

A evolução do consumo de memória, exibe quatro momentos importantes, representados na figura pelos pontos A, B, C e D. Os pontos A e B correspondem respectivamente ao lançamento da aplicação cliente do Infrastructure Node, e ao início da aplicação de informação estendida sobre produtos. O terceiro ponto (C), ocorre quando o utilizador captura a imagem do código de barras e inicia o processo de obtenção de informação de um produto. Finalmente, o quarto ponto (D) é relativo à terminação da aplicação de informação estendida sobre produtos.

De forma a permitir uma análise comparativa do perfil de ocupação de memória apresentado neste teste, avaliou-se a memória utilizada no contexto da visualização de uma página web. Para efeitos de teste utilizou-se o browser standard do dispositivo[1] e foi efectuada a visualização da página da CNN[2], verificando-se um gasto de memória de 1656 KB.

A aplicação cliente do Infrastructure Node, cujo arquivo jar tem 81,2 KB, apresenta um gasto de memória de 756 KB, o que é na ordem daquilo que é gasto na visualização de uma página web.

Já a aplicação de informação estendida sobre produtos apresenta um consumo de memória bastante superior, o que à partida é expectável já que esta faz uso de um conjunto substancial de recursos do telemóvel, em particular a sua câmara. O arquivo

[1] http://www.nokia.com/microsites/s60-browser-site
[2] http://www.cnn.com

jar desta aplicação, ocupa 308 KB e o seu consumo de memória, no ponto máximo é de 8640 KB.

6.3 Funcionalidade de execução remota

Como referido anteriormente, nesta secção avalia-se o benefício de executar parte duma aplicação na infra-estrutura, usando para o efeito a aplicação de informação estendida sobre produtos descrita na secção 5.2.3. Os resultados apresentados comparam o acesso à mesma informação no dispositivo móvel, acedendo directamente às fontes de dados ou transferindo parte da aplicação para a infra-estrutura.

Foram realizados dois testes distintos que mediram o tempo até à apresentação da informação e a quantidade de dados transferidos no processo. No primeiro teste, a informação acedida é relativa a um produto. No segundo teste, a informação acedida consiste num vídeo.

6.3.1 Teste 1

O objectivo do primeiro teste foi avaliar o tempo que demora a mostrar a informação relativa a um produto, quantificando os dados que foram transferidos para o dispositivo móvel e para a infra-estrutura. Esta informação consiste numa imagem do produto, o seu nome e respectiva classificação, uma listagem de outros produtos relacionados, bem como informação detalhada sobre o vídeo sugerido (título, duração e imagem) para além de uma lista de outros videos relacionados. Para obter esta informação são acedidos os serviços do YouTube e da Amazon.

Este teste foi efectuado em três cenários distintos. Primeiro, com todo o processamento a ser efectuado no dispositivo móvel. Seguidamente usando a infra-estrutura para contactar os diversos serviços e transferindo os resultados através de bluetooth e por último, usando também a infra-estrutura mas transferindo os dados via Wi-Fi. Os dados obtidos apresentam-se na tabela 6.3.

		Transferências (KB)	
	Tempo (ms)	**Infra-estrutura**	**Móvel**
Stand-alone	28441	0	144,087
Bluetooth	2442	147546	12,945
Wi-Fi	4671	147546	12,945

Tabela 6.3: Tempos médios de execução e quantidade de dados transferida no primeiro teste

Na utilização da infra-estrutura traduz-se numa redução significativa do tempo de resposta. Este facto deve-se à transferência de uma menor quantidade de dados, de 147

KB para 13 KB, para o dispositivo móvel quando a aplicação se encontra a funcionar em modo de infra-estrutura.

6.3.2 Teste 2

O segundo teste avalia os potenciais ganhos de se usar a infra-estrutura para efectuar a adaptação de conteúdos com alta qualidade, antes destes serem transferidos para o dispositivo móvel. Este teste foi realizado com base em duas experiências distintas. A primeira consistiu na utilização não modificada da aplicação de informação estendida sobre produtos, acedendo aos videos fornecidos pelo YouTube. A segunda consistiu na utilização de uma versão modificada desta aplicação de forma a aceder a conteúdos em alta definição.

Qualquer uma das experiências referidas foi efectuada em três cenários distintos. No primeiro, o vídeo é visualizado directamente no equipamento móvel. Nos seguintes varia a tecnologia de comunicação utilizada (bluetooth ou Wi-Fi), e o dispositivo móvel contacta a componente que executa na infra-estrutura para efectuar o download e adaptação do vídeo às características do equipamento, antes de efectuar a sua transferência para o cliente. A adaptação efectuada consistiu numa mudança de formato, para MPEG-4, e num redimensionamento dos conteúdos para a resolução de 352x288.

Os resultados relativos à execução em modo de infra-estrutura da versão não modificada da aplicação estão concentrados na tabela 6.4. Em modo stand-alone a aplicação acede directamente ao website do YouTube através do browser do dispositivo, fazendo *streaming* do vídeo e por isso não se trata de uma situação comparável com as anteriores. Nesta situação a aplicação demora 8620ms em média, até iniciar a reprodução do vídeo. Comparando este resultado com os resultados obtidos em modo de infra-estrutura, verifica-se que a utilização da infra-estrutura para efectuar a sua adaptação dos conteúdos não grandes vantagens neste contexto.

	Tempo (s)				**Transferências (MB)**		
	Total	**Download**	**Transcoding**	**Trans. p/ móvel**	**Total**	**Infra-estrutura**	**Móvel**
Bluetooth	**24,20**	4,9495	3,831	15,42	**2,27**	1,21	1,06
Wi-Fi	**15,18**	4,9495	3,831	6,40	**2,27**	1,21	1,06

Tabela 6.4: Tempos médios e quantidade de dados transferida para videos do YouTube

Na segunda experiência recorreu-se a uma versão modificada da aplicação de informação estendida sobre produtos. O objectivo principal foi testar a capacidade desta lidar com conteúdos com alta qualidade, avaliando os potenciais benefícios da sua execução parcial na infra-estrutura. Para tal em vez de recorrer aos videos fornecidos pelo YouTube,

foi efectuado o download de um vídeo em alta definição [3]. Na tabela 6.5 encontra-se o resumo dos dados obtidos.

	Tempo (s)				Transferências (MB)		
	Total	**Download**	**Transcoding**	**Trans. p/ móvel**	**Total**	**Infra-estrutura**	**Móvel**
Stand-alone	**351,81**	351,81	0,00	0,00	**107,91**	0,00	107,91
Bluetooth	**222,56**	54,88	98,21	69,47	**112,88**	107,91	4,97
Wi-Fi	**178,58**	54,88	98,21	25,49	**112,88**	107,91	4,97

Tabela 6.5: Tempos médios e quantidade de dados transferida para as várias entidades no segundo teste

Os resultados, confirmam que as comunicações para o telemóvel incorrem num overhead considerável. Como se pode verificar, para o download do mesmo ficheiro de cerca de 108 MB, o tempo de transferência na infra-estrutura é cerca de um quinto. Este teste evidencia também o potencial da utilização de um serviço de adaptação de conteúdos, já que neste caso concreto, foi possível transformar um vídeo com alta qualidade e com cerca de 108 MB, num vídeo em formato MPEG-4 com cerca de 5 MB, sendo obviamente a sua transferência para o dispositivo móvel muito mais rápida.

Em suma, os resultados obtidos vêm confirmar que a utilização da infra-estrutura para correr as componentes da aplicação mais exigentes em termos computacionais, ou de comunicação traz efectivamente vantagens. Essas vantagens traduzem-se numa redução bastante grande do tempo de resposta das aplicações e na redução do tráfego para o equipamento móvel. Este último facto pode contribuir para um menor consumo de energia, apesar de não ter sido possível testar este último aspecto.

[3] http://movies.apple.com/movies/summit/thehurtlocker/thehurtlocker_h720p.mov

7

Conclusão

Nesta dissertação apresentou-se uma plataforma ubíqua para fornecimento de serviços num espaço público, designada por PIPE (Pervasive Infrastructure for Public Spaces).

Um sistema desta natureza, que permita aos utilizadores equipados com dispositivos móveis fazer uso dos recursos fornecidos pela infra-estrutura, apresenta diversos desafios. O PIPE foi concebido para ter as seguintes características: baixo custo, tempo de resposta curto, i.e., na ordem dos segundos, flexibilidade e segurança.

Este sistema foi influenciado por outros trabalhos já efectuados no mesmo âmbito. À semelhança de sistemas como [LJP+06, RRA08], o PIPE oferece uma arquitectura orientada aos serviços. Esta abordagem permite definir e disponibilizar um conjunto de aplicações e serviços que podem integrar os recursos fornecidos pela infra-estrutura (e.g. ecrãs públicos, capacidade de processamento, etc), com os dispositivos pessoais dos utilizadores.

Ao contrário de sistemas desenvolvidos com objectivos semelhantes, o PIPE suporta a execução de aplicações definidas pelo utilizador na infra-estrutura. Esta funcionalidade, permite dotar o sistema de um maior dinamismo, tendo contudo, implicações de segurança. Em particular é necessário criar os mecanismos que permitam a protecção da plataforma face à execução de código arbitrário, e oferecer os mecanismos que permitam o controlo dos acessos das aplicações aos serviços disponibilizados.

Esta plataforma acomoda um espectro alargado de aplicações, desde aplicações que executam exclusivamente na infra-estrutura, até aquelas que executam nos dispositivos móveis dos utilizadores, e que podem eventualmente fazer uso de alguns serviços disponibilizados pelo sistema.

Entre estes dos extremos, existe o conjunto de aplicações que faz um uso mais extensivo das capacidades oferecidas pelos diferentes tipos de dispositivos. Nestas, a execução é distribuída entre a infra-estrutura e os equipamentos pessoais dos utilizadores, possibilitando que as componentes que exigem mais recursos (processamento, comunicação, memória, etc) possam correr na infra-estrutura, podendo as restantes componentes executar nos equipamentos móveis.

De forma a validar este sistema, foram criadas diferentes aplicações e serviços. Mais concretamente, foram criadas duas aplicações de teste. A primeira é uma aplicação de publicidade, que executa exclusivamente na infra-estrutura, não requerendo uma interacção explícita com os utilizadores ou os seus dispositivos. Esta aplicação demonstra a possibilidade de suportar aplicações que correm completamente na infra-estrutura, e que estão distribuídas por vários nós.

A segunda, é uma aplicação que permite obter informação adicional sobre um produto, baseado na captura do seu código de barras com a câmara do dispositivo móvel. Nesta, a informação adicional sobre o produto em causa é obtida através do contacto com diversas fontes de dados presentes na web. O módulo que é responsável pela agregação dessa informação, pode ser executado quer na infra-estrutura, quer no dispositivo móvel, dependendo da preferência do utilizador. Esta aplicação mostra que a plataforma permite a utilização integrada de dispositivos móveis, com recursos fornecidos pela infra-estrutura, de forma a suportar a execução distribuída de aplicações.

Os serviços implementados, dão suporte às aplicações descritas anteriormente. Estes exibem bastante diversidade, podendo ser serviços mais simples, como por exemplo de detecção de dispositivos bluetooth, ou de câmaras, até serviços mais complexos como detecção de faces e publicidade. A utilização dos serviços simplifica a criação de aplicações.

Os resultados obtidos mostram que a utilização combinada de dispositivos móveis, com os recursos fornecidos pela infra-estrutura, permite reduções significativas na quantidade de dados transferidos para o equipamento móvel. Esta redução traduz-se numa diminuição do tempo de resposta e numa potencial poupança de energia. Estes resultados são particularmente evidentes nas situações em que são manipulados conteúdos com alta qualidade.

7.1 Trabalho futuro

O trabalho apresentado neste documento, exibe diversos aspectos que podem ser abordados futuramente, de forma a oferecer uma solução mais completa.

- De forma a evitar ou identificar utilizações ilícitas dos recursos disponibilizados (e.g download de conteúdos ilegais), é interessante providenciar os mecanismos que permitam certificar ou auditar os resultados das computações efectuadas na infra-estrutura. Com base nestes mecanismos, seria eventualmente possível proceder à responsabilização dos utilizadores envolvidos.

- O controlo de acessos de uma aplicação relativamente aos serviços que pode utilizar é um aspecto importante de abordar neste sistema. Para lidar com este problema seria possível combinar a utilização do sistema *Kerberos* com um sistema de *capacidades*, em que um cliente apresentasse aos vários Infrastructure Nodes um *ticket* com as respectivas permissões.

- As aplicações que correm no PIPE podem forçar que determinados serviços só sejam utilizados se executarem no mesmo Infrastructure Node. Nos casos em que tal restrição não se coloca, o endereço do serviço que é dado à aplicação, corresponde ao primeiro serviço encontrado, independentemente do nó onde este se encontre. Este processo de escolha pode ser alvo de melhorias, sendo interessante por exemplo tentar distribuir a utilização dos serviços pelos nós que estejam sujeitos a uma carga menor.

- No protótipo actual, o processo pelo qual o utilizador contacta o Infrastructure Node não é muito prático. Um aspecto interessante a explorar prende-se com a utilização de diferentes técnicas alternativas para efectuar o emparelhamento com a infra-estrutura, por exemplo recorrendo à câmara do dispositivo móvel do utilizador.

- O acesso concorrente a certos recursos, em particular ecrãs, é uma questão fundamental para um sistema como o PIPE. No protótipo actual foi apresentada uma solução tentativa para esta questão, sob a forma de um serviço de reprodução de multimédia. Contudo, esta solução é claramente insuficiente para acomodar a riqueza de possíveis interacções que podem ocorrer. É por isso interessante explorar mecanismos mais robustos que permitam lidar com acessos concorrentes a este tipo de meios.

- No protótipo actual não são contemplados mecanismos que permitam limitar os recursos (e.g. tempo de CPU, memória, etc.) que uma aplicação ou serviço utilizam. Numa solução mais completa este é um aspecto importante de abordar, sendo interessante explorar a integração de tecnologias de virtualização ao nível do sistema de operação (e.g. OpenVZ), com uma arquitectura orientada aos serviços. Este tipo de virtualização permite a instanciação mais rápida de

máquinas virtuais relativamente a soluções de virtualização completa (e.g. VirtualBox), o que é importante no contexto onde um sistema como o PIPE pode ser implementado. Isto permitiria não só resolver a questão da alocação de recursos, mas também daria a possibilidade do utilizador executar aplicações feitas noutras linguagens, que não Java.

Anexo

A.1 Ficheiro de configuração de uma aplicação

Listagem A.1: XML Schema do ficheiro de configuração de uma aplicação

```
1  <?xml version="1.0" encoding="UTF-8"?> <xs:schema
2  xmlns:xs="http://www.w3.org/2001/XMLSchema"> <xs:element name="application"
       >
3          <xs:complexType>
4              <xs:sequence>
5                  <xs:element name="name" type="xs:string" />
6                  <xs:element name="jarFile" type="xs:string" />
7                  <xs:element name="confFile" type="xs:string" />
8                  <xs:element name="description" type="xs:string" minOccurs="
                       0"
9                  maxOccurs="1"/>
10                 <xs:element name="remoteFileType" type="locationType"
                       minOccurs="0" maxOccurs="1"/> <xs:element name="
                       entryPoint" type="xs:string" />
11                 <xs:element name="applicationArguments" type="xs:string"
                       minOccurs="0" maxOccurs="1"/>
12                 <xs:element name="interfaces" type="interfaceType"/>
13                 <xs:element name="services" type="servicesList"/>
14             </xs:sequence>
15         </xs:complexType>
16     </xs:element>
17
18     <xs:complexType name="interfaceType">
```

```
19          <xs:sequence>
20              <xs:element name="web" type="webInterfaceType" minOccurs="0"
21                  maxOccurs="1" />
22              <xs:element name="j2me" type="j2meInterfaceType"
23                  minOccurs="0" maxOccurs="1" />
24          </xs:sequence>
25      </xs:complexType>
26
27      <xs:complexType name="servicesList">
28          <xs:sequence>
29              <xs:element name="service" type="serviceType" minOccurs="0"/>
30          </xs:sequence>
31      </xs:complexType>
32
33      <xs:complexType name="webInterfaceType">
34          <xs:sequence>
35              <xs:element name="webEntryPoint" type="xs:string" />
36          </xs:sequence>
37      </xs:complexType>
38
39      <xs:complexType name="j2meInterfaceType">
40          <xs:sequence>
41              <xs:element name="j2meUrl" type="xs:string" />
42              <xs:element name="initUrl" type="xs:string" />
43          </xs:sequence>
44      </xs:complexType>
45
46      <xs:simpleType name="locationType">
47          <xs:restriction base="xs:string">
48              <xs:enumeration value="LOCAL"/>
49              <xs:enumeration value="WEB"/>
50          </xs:restriction>
51      </xs:simpleType>
52
53      <xs:complexType name="serviceType">
54          <xs:sequence>
55              <xs:element name="serviceName" type="xs:string"/>
56              <xs:element name="jarFile" type="xs:string" />
57              <xs:element name="confFile" type="xs:string" />
58              <xs:element name="serviceLocation" type="xs:string" />
59          </xs:sequence>
60      </xs:complexType>
61
62  </xs:schema>
```

A.2 Ficheiro de configuração do *Infrastructure Node*

Listagem A.2: DTD do ficheiro de configuração do Infrastructure Node

```
<!ELEMENT infrastructureNode (adminInfo, services, applications)>
<!ELEMENT adminInfo (organization, nodeName, path, serviceAddress,
    pairingAddress, tcpPairingPort, httpServerPort, uddiUrl, publisherId,
    serviceInstallDir)>
<!ELEMENT services (service*)>
<!ELEMENT applications (application*)>
<!ELEMENT service (name, serviceJarFile, serviceConfFile)>
<!ELEMENT server (serviceName, handler, serviceParams)>
<!ELEMENT client (connectionParams, localPort, handler)>
<!ELEMENT application (name, jarFile, confFile)>

<!ELEMENT organization (#PCDATA)>
<!ELEMENT nodeName (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT serviceAddress (#PCDATA)>
<!ELEMENT pairingAddress (#PCDATA)>
<!ELEMENT tcpPairingPort (#PCDATA)>
<!ELEMENT httpServerPort (#PCDATA)>
<!ELEMENT uddiUrl (#PCDATA)>
<!ELEMENT publisherId (#PCDATA)>
<!ELEMENT serviceInstallDir (#PCDATA)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT handler (#PCDATA)>
<!ELEMENT remote (#PCDATA)>
<!ELEMENT serviceParams (#PCDATA)>
<!ELEMENT connectionParams (#PCDATA)>
<!ELEMENT localPort (#PCDATA)>
<!ELEMENT serviceName (#PCDATA)>

<!ELEMENT serviceJarFile (#PCDATA)>
<!ELEMENT serviceConfFile (#PCDATA)>

<!ELEMENT jarFile (#PCDATA)>
<!ELEMENT confFile (#PCDATA)>
```

Bibliografia

[AGKO04] Lauri Aalto, Nicklas Göthlin, Jani Korhonen, and Timo Ojala. Bluetooth and wap push based location-aware mobile advertising system. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 49–58, New York, NY, USA, 2004. ACM.

[BFS07] Joao Barreto, Paulo Ferreira, and Marc Shapiro. Exploiting our computational surroundings for better mobile collaboration. *Mobile Data Management, IEEE International Conference on*, 0:110–117, 2007.

[BGSH07] Rajesh Krishna Balan, Darren Gergle, Mahadev Satyanarayanan, and James Herbsleb. Simplifying cyber foraging for mobile devices. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 272–285, New York, NY, USA, 2007. ACM.

[BIF+04] Harry Brignull, Shahram Izadi, Geraldine Fitzpatrick, Yvonne Rogers, and Tom Rodden. The introduction of a shared interactive surface into a communal space. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 49–58, New York, NY, USA, 2004. ACM.

[BJA98] Harini Bharadvaj, Anupam Joshi, and Sansanee Auephanwiriyakul. An active transcoding proxy to support mobile web access. In *SRDS '98: Proceedings of the The 17th IEEE Symposium on Reliable Distributed Systems*, page 118, Washington, DC, USA, 1998. IEEE Computer Society.

[BKN+05] Stefan Berger, Rick Kjeldsen, Chandra Narayanaswami, Claudio Pinhanez, Mark Podlaseck, and Mandayam Raghunath. Using symbiotic displays to view sensitive information in public. In *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 139–148, Washington, DC, USA, 2005. IEEE Computer Society.

[blu09] Bluetooth.com - specification documents, 23 Jul. 2009. `http://bluetooth.com/Bluetooth/Technology/Building/Specifications/Default.htm`.

[BSSW02] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *In Symposium on Network and Distributed Systems Security (NDSS 02*, 2002.

[CBC+08] Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, Dirk Haehnel, Beverly Harrison, Bruce Hemingway, Jeffrey Hightower, Pedja Klasnja, Karl Koscher, Anthony LaMarca, Jonathan Lester, James Landay, Louis Legrand, Ali Rahimi, Adam Rea, and Danny Wyatt. The mobile sensing platform: An embedded system for capturing and recognizing human activities. *IEEE Pervasive Computing*, 7(2):32–41, April-June 2008.

[CKM+08] Sunny Consolvo, Predrag Klasnja, David W. McDonald, Daniel Avrahami, Jon Froehlich, Louis LeGrand, Ryan Libby, Keith Mosher, and James A. Landay. Flowers or a robot army?: encouraging awareness & activity with personal, mobile displays. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 54–63, New York, NY, USA, 2008. ACM.

[DKC05] Jean Dollimore, Tim Kindberg, and George Coulouris. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science Series)*. Addison Wesley, 2005.

[DSP08] YeoSong Moon Scott McFaddin Chandra Narayanaswami HyunKi Jang Daniel Coffman MyungChul Lee JongKwon Lee Danny Soroker, YoungSang Paik and Jinwoo Park. User-defined mashups in interactive public spaces. In *Proc 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2008)*, 2008.

[GC04] Sachin Goyal and John Carter. A lightweight secure cyber foraging infrastructure for resource-constrained devices. In *WMCSA '04: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 186–195, Washington, DC, USA, 2004. IEEE Computer Society.

[GCB+08] Scott Garriss, Rámon Cáceres, Stefan Berger, Reiner Sailer, Leendert van Doorn, and Xiaolan Zhang. Trustworthy and personalized computing on public kiosks. In *MobiSys '08: Proceeding of the 6th international conference*

on Mobile systems, applications, and services, pages 199–210, New York, NY, USA, 2008. ACM.

[GJC09] Filipe Grangeiro, Rui Jesus, and Nuno Correia. Face recognition and gender classification in personal memories. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 0:1945–1948, 2009.

[gsm09] 3gpp - specifications, 23 Jul. 2009. `http://www.3gpp.org/Specifications`.

[HKB08] Elaine M. Huang, Anna Koster, and Jan Borchers. Overcoming assumptions and uncovering practices: When does the public really look at public displays? In *Pervasive 2008*, volume 5013/2008 of *Lecture Notes in Computer Science*, pages 228–243, Sydney, Australia, May 2008. Springer Link.

[ird09] Irda, 23 Jul. 2009. `http://www.irda.org/`.

[JP06] Rui José and Helder Pinto. Display-centred applications in ubiquitous computing. *Ubicomp 2006 : proceedings of the 8th Annual Conference on Ubiquitous Computing : proceedings*, 2006.

[KBM+00] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, and Bill Serra. People, places, things: web presence for the real world. In *In proceedings WMCSA2000. Available as http://www.cooltown.hp.com/papers/webpres/webpresence.htm*, pages 365–376, 2000.

[KPD07] M. Karam, T. Payne, and E. David. Evaluating bluscreen: Usability for intelligent pervasive displays. *Pervasive Computing and Applications, 2007. ICPCA 2007. 2nd International Conference on*, pages 18–23, July 2007.

[LJP+06] MC Lee, HK Jang, YS Paik, SE Jin, and S Lee. Ubiquitous device collaboration infrastructure: Celadon. In *SEUS-WCCIA '06: Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, pages 141–146, Washington, DC, USA, 2006. IEEE Computer Society.

[MC07] Rui Neves Madeira and Nuno Correia. Interaction between shared displays and mobile devices in an augmented objects framework. In *UBICOMM '07: Proceedings of the International Conference on Mobile Ubiquitous*

Computing, Systems, Services and Technologies, pages 189–194, Washington, DC, USA, 2007. IEEE Computer Society.

[mob09] Itu information and communication technology, 23 Jul. 2009. `http://www.itu.int/ITU-D/ict/statistics/`.

[NC05] ROSU M C JANG H K JIN S E KIM S Y LEE M C LEE S PAIK Y S NARAYANASWAMI C, RAGHUNATH M T. Device collaboration for ubiquitous computing: Scenarios and challenges. pages 7–12, 2005.

[NCL+08] C. Narayanaswami, D. Coffman, M. C. Lee, Y. S. Moon, J. H. Han, H. K. Jang, S. McFaddin, Y. S. Paik, J. H. Kim, J. K Lee, J. W. Park, and D. Soroker. Pervasive symbiotic advertising. In *HotMobile '08: Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 80–85, New York, NY, USA, 2008. ACM.

[nfc09] Standard ecma-340 near field communication interface and protocol, 23 Jul. 2009. `http://www.ecma-international.org/publications/standards/Ecma-340.htm`.

[NSN+97] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. *SIGOPS Oper. Syst. Rev.*, 31(5):276–287, 1997.

[obe09] IrDA Specifications and Technical Notes, 23 Jul. 2009. `http://www.irda.org/displaycommon.cfm?an=1&subarticlenbr=7`.

[PAB+04] Tim Paek, Maneesh Agrawala, Sumit Basu, Steve Drucker, Trausti Kristjansson, Ron Logan, Kentaro Toyama, and Andy Wilson. Toward universal mobile interaction for shared displays. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 266–269, New York, NY, USA, 2004. ACM.

[PDJS06] Terry Payne, Ester David, Nicholas R. Jennings, and Matthew Sharifi. Auction mechanisms for efficient advertisement selection on public displays. In *Proceedings of European Conference on Artificial Intelligence*, pages 285–289, 2006.

[PKH+06] Shwetak N. Patel, Julie A. Kientz, Gillian R. Hayes, Sooraj Bhat, and Gregory D. Abowd. Farther than you may think: An empirical investigation of the proximity of users to their mobile phones. In *in Proceedings of UbiComp 2006: Ubiquitous Computing, P. Dourish*, pages 123–140. Springer, 2006.

[R-O09] R-OSGi - transparent OSGi remote extension for distributed services - R-OSGi, 17 Jul. 2009. http://r-osgi.sourceforge.net/.

[RC02] Anand Ranganathan and Roy H. Campbell. Advertising in a pervasive computing environment. In *WMC '02: Proceedings of the 2nd international workshop on Mobile commerce*, pages 10–14, New York, NY, USA, 2002. ACM.

[RRA08] Jan S. Rellermeyer, Oriana Riva, and Gustavo Alonso. Alfredo: an architecture for flexible interaction with electronic devices. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 22–41, New York, NY, USA, 2008. Springer-Verlag New York, Inc.

[Sat01] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 8(4):10–17, 2001.

[SF05] Ya-Yunn Su and Jason Flinn. Slingshot: deploying stateful services in wireless hotspots. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 79–92, New York, NY, USA, 2005. ACM.

[SOA09] Soap - specifications, 23 Jul. 2009. http://www.w3.org/TR/soap/.

[SPD06] Matthew Sharifi, Terry Payne, and Esther David. Public display advertising based on bluetooth device presence. In *Mobile Interaction with the Real World (MIRW 2006) in conjunction with the 8th International Conference on Human Computer Interaction with Mobile Devices and Services*, Espoo, Finland, September 2006.

[TSN07] Ville Tuulos, Jürgen Scheible, and Heli Nyholm. Combining web, mobile phones and public displays in large-scale: Manhattan story mashup. *Pervasive Computing*, pages 37–54, 2007.

[UDD09] OASIS - Committees - OASIS UDDI Specifications TC, 9 Jun. 2009. http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm.

[Wei99] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999.

[wif09] Ieee-sa getieee 802.11 lan/man wireless lans, 23 Jul. 2009. http://standards.ieee.org/getieee802/802.11.html.

[wsd09] Web services description language (wsdl), 21 Jul. 2009. http://www.w3.org/TR/wsdl.